Mask Transfiner for High-Quality Instance Segmentation

Lei Ke[1,2] Martin Danelljan[1] Xia Li[1] Yu-Wing Tai[3] Chi-Keung Tang[2] Fisher Yu[1]
[1]ETH Zürich [2]HKUST [3]Kuaishou Technology

Abstract

Two-stage and query-based instance segmentation methods have achieved remarkable results. However, their segmented masks are still very coarse. In this paper, we present Mask Transfiner *for high-quality and efficient instance segmentation. Instead of operating on regular dense tensors, our Mask Transfiner decomposes and represents the image regions as a quadtree. Our transformer-based approach only processes detected error-prone tree nodes and self-corrects their errors in parallel. While these sparse pixels only constitute a small proportion of the total number, they are critical to the final mask quality. This allows Mask Transfiner to predict highly accurate instance masks, at a low computational cost. Extensive experiments demonstrate that Mask Transfiner outperforms current instance segmentation methods on three popular benchmarks, significantly improving both two-stage and query-based frameworks by a large margin of +3.0 mask AP on COCO and BDD100K, and +6.6 boundary AP on Cityscapes. Our code and trained models are available at* https://github.com/SysCV/transfiner.

1. Introduction

Advancements in image instance segmentation has largely been driven by the developments of powerful object detection paradigms. Approaches based on Mask R-CNN [12,21,24,28,34] and more recently DETR [15,17,23] have achieved ever increasing performance on, for instance, the COCO challenge [33]. While these methods excel in detection and localization of objects, the problem of efficiently predicting highly accurate segmentation masks has so far remained elusive.

As shown in Figure 3, there is still a significant gap between the bounding box and segmentation performance of the recent state-of-the-art methods, especially for the recent query-based methods. This strongly indicates that improvements in mask quality has not kept pace with the advancements detection capability. In Figure 2, the predicted masks of previous methods are very coarse, most often oversmoothing object boundaries. In fact, efficient and accurate mask prediction is highly challenging, due to the need for high-resolution deep features, which demands large computational and memory costs [38].

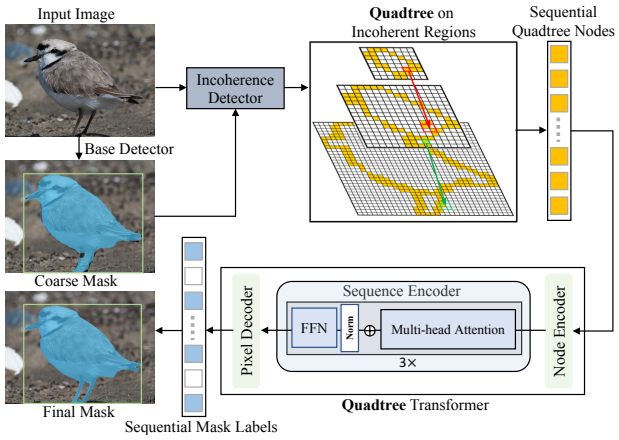


Figure 1. We propose Mask Transfiner for high-quality instance segmentation. It first builds a quadtree based on the sparse incoherent regions on the RoI pyramid and then jointly refines all tree nodes using the refinement transformer with quadtree attention.

To address these issues, we propose Mask Transfiner, an efficient transformer-based approach for high-quality instance segmentation. In Figure 1, our approach first identifies error-prone regions, which are mostly strewn along object boundaries or in high-frequency regions. To this end, our network learns to detect *incoherent regions*, defined by the loss of information when downsampling mask itself. These incoherent pixels are sparsely located, consisting only of a small portion of the total pixels. However, as they are shown to be critical to the final segmentation performance, it allows us to only process small parts of the high-resolution feature maps in the refinement process. Thus, we build a hierarchical quadtree [18] to represent and process the incoherent image pixels at multiple scales.

To refine the mask labels of the incoherent quadtree nodes, we design an refinement network based on the transformer instead of standard convolutional networks because they require operating on uniform grids. Our transformer has three modules: node encoder, sequence encoder and pixel decoder. The node encoder first enriches the feature embedding for each incoherent point. The sequence encoder then takes these encoded feature vectors across multiple quadtree levels as input queries. Finally, the pixel decoder predicts their corresponding mask labels. Comparing

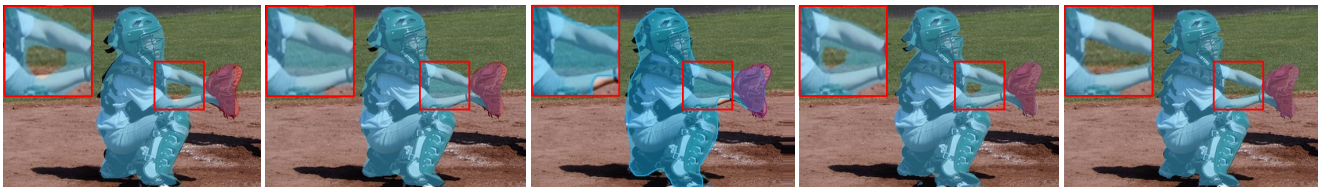


Figure 2. Instance Segmentation on COCO [33] validation set by a) Mask R-CNN [21], b) BMask R-CNN [12], c) SOLQ [15], d) PointRend [28], g) Mask Transfiner (Ours) using R50-FPN as backbone, where Mask Transfiner produces significantly more detailed results at high-frequency image regions by replacing Mask R-CNN's default mask head. Zoom in for better view.

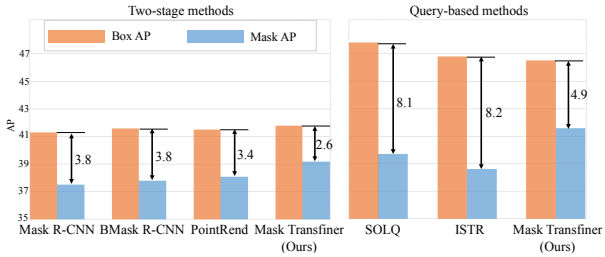


Figure 3. The performance gap between object detection and segmentation for instance segmentation models on COCO *test-dev* set using R50-FPN as backbone. Detailed comparisons are in Table 9.

to MLP [28], the sequential representation and multi-head attention enables Mask Transfiner to flexibly takes as input sparse feature points across levels in parallel, models their pixel-wise relations, and then propagates information among them even in a long distance range.

We extensively analyze our approach on COCO, Cityscapes and BDD100K benchmarks, where quantitative and qualitative results show that Mask Transfiner not only outperforms existing two-stage and query-based methods, but also is efficient in computation and memory cost compared to standard transformer usages. We establish a new state-of-the-art result on COCO *test-dev* of 41.6 AP^{Mask} using ResNet-50, outperforming most recent SOLQ [15] and QueryInst [17] by a significant margin.

2. Related Work

Instance Segmentation Two-stage instance segmentation methods [2, 6, 8, 12, 16, 21, 24, 31] first detects bounding boxes and then performing segmentation in each RoI region. Mask R-CNN [21] extends Faster R-CNN [35] with an FCN branch. The follow-up works [7, 12, 25, 26] also contribute to the family of Mask R-CNN models. One-stage methods [5, 8, 29, 30] and kernel-based method [48], such as PolarMask [44], YOLOACT [1], and SOLO [40, 41] remove the proposal generation and feature re-pooling steps, achieving comparable results with higher efficiency.

Query-based instance segmentation methods [15, 17, 19, 23, 42], which are inspired by DETR [4], have emerged very recently by treating segmentation as a set prediction problem. These methods use queries to represent the interested objects and jointly perform classification, detection and mask regression on them. In [15, 23], the object masks are compressed as encoding vectors using DCT or PCA algorithms, while QueryInst [17] adopts dynamic mask heads with mask information flow. However, the large gaps between the detection and segmentation performance in Figure 3 reveals that the mask quality produced by these query-based methods are still unsatisfactory. In contrast to the above methods, Mask Transfiner is targeted for high-quality instance segmentation. In our efficient transformer the input queries are incoherent pixels nodes, instead of representing the objects. Our method is applicable to and effective in both the two-stage and query-based frameworks.

Refinement for Instance Segmentation Most existing works on instance segmentation refinement rely on specially designed convolutional networks [36, 47] or MLPs [28]. PointRend [28] samples feature points with low-confidence scores and refines their labels with a shared MLP, where the selected points are determined by the coarse predictions of the Mask R-CNN. RefineMask [47] incorporates fine-grained features with an additional semantic head as the guidance. The post-processing method BPR [36] crops boundary patches of images and initial masks as input and use [38] for segmentation. Notably some methods [11, 14, 46] focus on refining semantic segmentation details. However, it is challenging for instance segmentation due to the more complex segmentation setting, with varying number of objects per image and the requirement of delineating overlapping objects [27].

Compared to these refinement methods, Mask Transfiner is an end-to-end instance segmentation method, using a transformer for correcting errors. The regions to be refined are predicted using a lightweight FCN, instead of non-deterministic sampling based on mask scores [28]. Different from the MLP in [28], the sequential and hierarchical input representation enables Mask Transfiner to efficiently take non-local sparse feature points as input queries, where the strong global processing of transformers is a natural fit for our quadtree structure.

3. Mask Transfiner

We propose an approach to efficiently tackle high-quality instance segmentation. The overall architecture of Mask Transfiner is depicted in Figure 5. From the base object detection network, *e.g.* Mask R-CNN [21], we employ a multi-scale deep feature pyramid. The object detection head then predicts bounding boxes as instance proposals. This com-

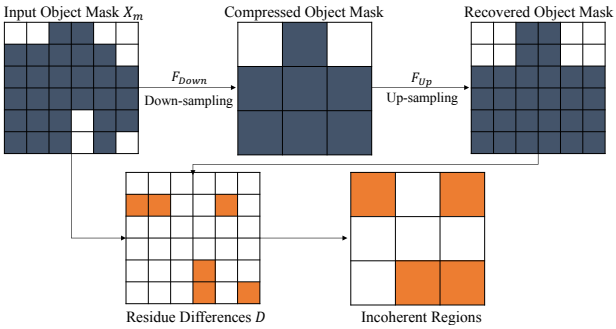


Figure 4. Illustration on incoherent regions definition by simulating mask information loss.

ponent also generates a coarse initial mask prediction at low resolution. Given this input data, our aim is to predict highly accurate instance segmentation masks.

Since much of the segmentation errors are attributed to the loss of spatial resolution, we first define such *incoherent regions* and analyze their properties in Section 3.1. To identify and refine incoherent regions in multiple scales, we employ a quadtree, discussed in Section 3.2. The lightweight incoherent region detector takes as input the coarse initial mask alongside the multi-scale features, and predicts the incoherent regions for each scale in a cascaded manner. This allows ours Mask Transfiner to save huge computational and memory burdens, because only a small part of the high-resolution image features are processed by the refinement network itself. Our refinement transformer, detailed in Section 3.3, operates in the detected incoherent regions. Since it operates on feature points on the constructed quadtree, and not in a uniform grid, we design a transformer architecture which jointly processes all incoherent nodes in all levels of the quadtree. Finally, we present the training strategy of Mask Transfiner along with the implementation details.

3.1. Incoherent Regions

Much of the segmentation errors produced by existing instance segmentation methods [15, 21] are due to the loss of spatial resolution, such as the mask downsampling operations, small RoI pooling size, and coefficients compression [15, 23], where mask prediction itself is performed at a coarse feature scale. Despite its efficiency, low spatial resolution makes it challenging to predict accurate object boundaries, due to the loss of high-frequency details. In this section, we first define *incoherent regions*, where mask information is lost due to reduced spatial resolution. Then, by analyzing their properties, we observe that a large portion of the errors are indeed located in these regions.

Definition of Incoherent Regions To identify incoherent regions, we simulate the loss of information due to downsampling in the network by also downsampling the mask itself. Specifically, information is lost in regions where the mask cannot be correctly reconstructed by a subsequent upsampling step, as illustrated in Figure 4. Formally, let M_l be a binary ground-truth instance mask of an object at scale level l. The resolution at each scale level differs by a factor of 2, where $l = 0$ is the finest and $l = L$ is the coarsest scale. We denote $2\times$ nearest neighbor down and upsampling by $\mathcal{S}_\downarrow$ and $\mathcal{S}_\uparrow$ respectively. The incoherent region at scale l is then the binary mask achieved as,

$$D_l = \mathcal{O}_\downarrow(M_{l-1} \oplus \mathcal{S}_\uparrow(\mathcal{S}_\downarrow(M_{l-1}))) \,. \tag{1}$$

Here, $\oplus$ denotes the logical 'exclusive or' operation and $\mathcal{O}_\downarrow$ is $2\times$ downsampling by performing the logical 'or' operation in each 2×2 neighborhood. A pixel (x, y) is thus incoherent $D_l(x, y) = 1$ if the original mask value M_{l-1} differs from its reconstruction in at least one pixel in the finer scale level. Intuitively, incoherent regions are mostly strewn along object instance boundaries or high-frequency regions, consisting of points with missing or extra predicted wrong labels by coarse masks. We provide the visualizations of them in Figure 6 and Supp. file, which are sparsely and non-contiguously distributed on a typical image.

Table 1. Experimental analysis of the incoherent regions on COCO *val* set. Percent denotes the area ratio of incoherent regions in the object bounding boxes. $\text{Recall}_{\text{Err}}$ is the ratio for all wrongly predicted pixels per object. Acc is the accuracy rate for coarse mask predictions inside incoherent regions. $\text{AP}_{\text{Coarse}}$ is measured by using coarse mask predictions for whole object regions while AP_{GT} only fills the incoherent regions with the ground truth labels.

Percent	$\text{Recall}_{\text{Err}}$	Acc	AP_{GT}	$\text{AP}_{\text{Coarse}}$
14%	43%	56%	51.0	35.5

Properties of Incoherent Regions In Table 1, we provide an analysis of the incoherent regions defined above. It shows that a large portion of prediction errors are concentrated in these incoherent regions, occupying 43% of all wrongly predicted pixels, while only taking 14% to the corresponding bounding box areas. The accuracy of the coarse mask prediction in incoherent regions is 56%. By fixing the bounding boxes detector, we conduct an oracle study to fill all these incoherent regions for each object with ground truth labels, while leaving the remaining parts as initial mask predictions. Compared to using initial mask predictions in the incoherent regions, the performance surges from 35.5 AP to 51.0 AP, indeed justifying they are critical for improving final performance.

3.2. Quadtree for Mask Refinement

In this section, we describe our approach for detecting and refining incoherent regions in the image. Our approach is based on the idea of iteratively detecting and dividing the incoherent regions in each feature scale. By only splitting the identified incoherent pixels for further refinement, our approach efficiently processes high-resolution features by only focusing on the important regions. To formalize

our approach, we employ a quadtree structure to first identify incoherent regions across scales. We then predict the refined segmentation labels for all incoherent nodes in the quadtree, using our network detailed in Section 3.3. Finally, our quadtree is employed to fuse the new predictions from multiple scales by propagating the corrected mask probabilities from coarse to finer scales.

Detection of Incoherent Regions The right part of Figure 5 depicts the design of our lightweight module to efficiently detect incoherent regions on a multi-scale feature pyramid. Following a cascaded design, we first concatenate the smallest features and coarse object mask predictions as input, and use a simple fully convolutional network (four 3×3 Convs) followed by a binary classifier to predict the coarsest incoherence masks. Then, the detected lower-resolution masks are upsampled and fused with the larger-resolution feature in neighboring level to guide the finer incoherence predictions, where only single 1×1 convolution layer is employed. During training, we enforce the groundtruth incoherent points in lower-level generated by Eq. 1 within the coverage of their parent points in higher-level feature map.

Quadtree Definition and Construction We define a *point quadtree* for decomposing the detected incoherent regions. Our structure is illustrated in Figure 5, where one yellow point in higher-level of FPN feature (such as feature resolution 28 $\times$28) has four quadrant points in its neighboring lower-level FPN feature map (such as resolution 56$\times$56). These are all feature points but with different granularities because they are on different pyramid levels. In contrast to the conventional quadtree 'cells' used in computer graphics, where a quadtree 'cell' can have multiple points, the subdivision unit for our point quadtree is always on a single point, with the division of points decided by the detected incoherent values and the threshold for the binary classifier.

Based on the detected incoherent points, we construct a multi-level hierarchical quadtree, beginning from using the detected points in the highest-level feature map as root nodes. These root nodes are selected for subdividing to their four quadrants on the lower-level feature map, with larger resolution and more local details. Note that at the fine level, only the quadrant points detected as incoherent could make a further break down and the expansion of incoherent tree nodes is restricted in regions corresponding to the incoherent predictions at the previous coarse level.

Quadtree Refinement We refine the mask predictions of the incoherent nodes of the quadtree using a transformer-based architecture. Our design is described in Sec. 3.3. It directly operates on the nodes of the quadtree, jointly providing refined mask probabilities at each incoherent node.

Quadtree Propagation Given the refined mask predictions, we design a hierarchical mask propagation scheme that exploits our quadtree structure. Given the initial coarse masks predictions in low-resolution, Mask Transfiner first corrects the points labels belong to the root level of the quadtree, and then propagates these corrected point labels to their corresponding four quadrants in neighboring finer level by nearest neighbor interpolation. The process of labels correction is efficiently conducted on the incoherent nodes in a level-wise manner until reaching the finest quadtree level. Comparing to only correcting the labels of finest leaf nodes on the quadtree, it enlarges the refinement areas with negligible cost by propagating refinement labeled to leaf nodes of the intermediate tree levels.

3.3. Mask Transfiner Architecture

In this section, we describe the architecture of the refinement network, which takes as input the incoherent points on the built quadtree (Section 3.2) for final segmentation refinement. These points are sparsely distributed along the high-frequency regions across levels and not spatially contiguous. Thus, standard convolutional networks operating on uniform grids are not suitable. Instead, we design a refinement transformer, Mask Transfiner, that corrects the predictions of all incoherent quadtree nodes in parallel.

Accurately segmenting ambiguous points requires both fine-grained deep features and coarse semantic information. The network therefore needs strong modeling power to sufficiently relate points and their surrounding context, including both spatial and cross-level neighboring points. Thus, a transformer, which can take sequential input and perform powerful local and non-local reasoning through the multi-head attention layers, is a natural choice for our Mask Transfiner design. Compared to the MLP in [28], the strong global processing of transformers is a natural fit for our quadtree structure. It benefits the effective fusion of the multi-level feature points information with different granularities and the explicit modeling of pairwise point relations.

Figure 5 shows the overall architecture of our Mask Transfiner. Based on the hierarchical FPN [32], instance segmentation is tackled in a multi-level and coarse-to-fine manner. Instead of using single-level FPN feature for each object [21], Mask Transfiner takes as input sequence the sparsely detected feature points in incoherent image regions across the RoI feature pyramid levels, and outputs the corresponding segmentation labels.

RoI Feature Pyramid Given an input image, the CNN backbone network equipped with FPN first extracts hierarchical feature maps for downstream processing, where we utilize feature levels from P_2 to P_5. The base object detector [15, 21] predicts bounding boxes as instance proposals. Then the RoI feature pyramid is built by extracting RoI features across three different levels $\{P_i, P_{i-1}, P_{i-2}\}$ of FPN with increasing square sizes $\{28, 56, 112\}$. The starting level i is computed as $i = \left\lfloor i_0 + \log_2(\sqrt{WH}/224) \right\rfloor$, where $i_0 = 4$, W and H are the RoI width and height. The coarsest level features contain more contextual and semantic infor-

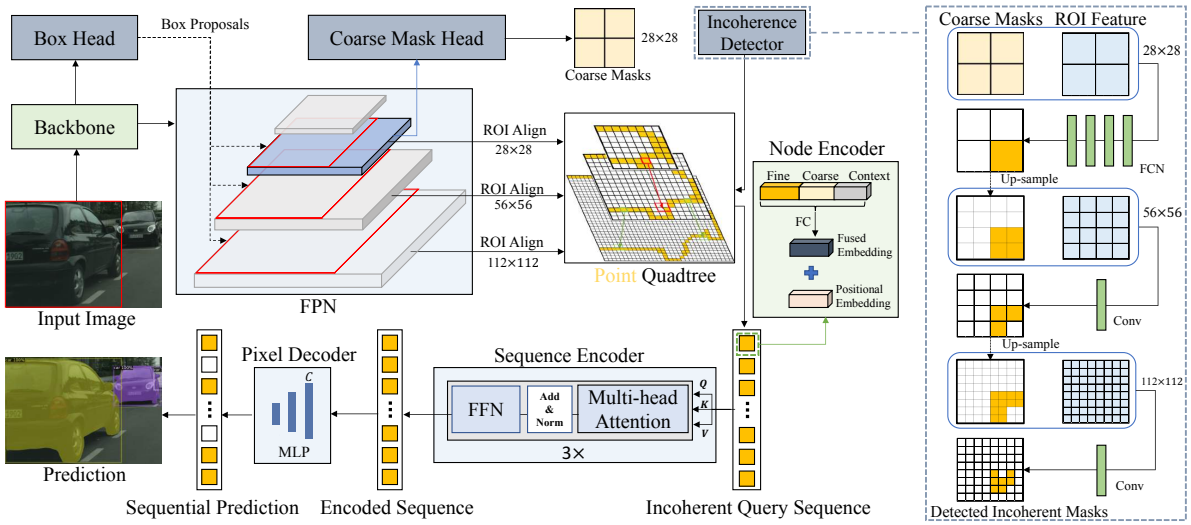


Figure 5. The framework of Mask Transfiner. On the point quadtree, yellow point grids denote detected incoherent regions requiring further subdivision to four quadrants. The incoherent query sequence is composed of points across three levels of the quadtree for joint refinement. The encoder of Transfiner consists of node encoder and sequence encoder, while the pixel decoder is on top of each self-attended query pixel and output their final labels. The incoherence detector is detailed in the right part of the figure with detections on multi-level incoherent regions (Yellow). The higher-resolution detection is under the guidance of the predicted incoherent mask up-sampled from lower level.

mation, while the finer levels resolve more local details.

Input Node Sequence Given the quadtree discussed in Section 3.2 along with the associated FPN features for each node, we construct the input sequence for our transformer-based architecture. The sequence consists of all incoherent nodes from all three levels of the quadtree. The resulting sequence thus has a size of $C \times N$, where N is the total number of nodes and C is the feature channel dimension. Notably, $N \ll HW$ due to the high degree of sparsity. Moreover, the ordering of the sequence does not matter due to the permutation invariance of transformer. In contrast to standard transformer encoder, the encoder of Transfiner has two parts: the node encoder and the sequence encoder.

Node Encoder To enrich the incoherent points feature, the node encoder of Mask Transfiner encodes each quadtree node using the following four different information cues: **1)** The fined-grained features extracted from corresponding location and level of the FPN pyramid. **2)** The initial coarse mask prediction from the base detector provides region-specific and semantic information. **3)** The relative positional encoding in each RoI encapsulates spatial distances and relations between nodes, capturing important local dependence and correlations. **4)** The surrounding context for each node captures local details to enrich the information. For each node, we use features extracted from the 3×3 neighborhood, compressed by a fully connected layer. Intuitively, this helps in localizing edges and boundaries, as well as capturing the local shape of the object. As illustrated in Figure 5, the fine-grained features, coarse segmentation cues and context features are first concatenated and fused by a FC layer to original feature dimension. The positional embedding is then added to the resulting feature vector.

Sequence Encoder and Pixel Decoder Then, the sequence transformer encoder of Transfiner jointly processes the encoded nodes from all levels in the quadtree. The transformer thus performs both global spatial and inter-scale reasoning. Each sequence encoder layer has a standard transformer structure, formed by a multi-head self-attention module and a fully connected feed forward network (FFN). To equip the incoherent points sequence with adequate positive and negative references, we also use all feature points from the coarsest FPN level with small size 14×14. Different from the standard transformer decoder [4] with deep attention layers, the pixel decoder in Mask Transfiner is a small two-layer MLP, which decodes the output query for each node in the tree, in order to predict the final mask labels.

Training and inference Based on the constructed quadtree, we develop flexible and adaptive training and inference schemes for Mask Transfiner, where all detected incoherent nodes across quadtree levels are formed into a sequence for parallel prediction. During inference, to obtain final object masks, Mask Transfiner follows the quadtree propagation scheme (Section 3.2) after obtaining the refined labels for incoherent nodes. During training, the whole Mask Transfiner framework can be trained in an end-to-end manner. We employ a multi-task loss,

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{Detect}} + \lambda_2 \mathcal{L}_{\text{Coarse}} + \lambda_3 \mathcal{L}_{\text{Refine}} + \lambda_4 \mathcal{L}_{\text{Inc}} \,. \quad (2)$$

Here, $\mathcal{L}_{\text{Refine}}$ denotes the refinement with L1 loss between the predicted labels for incoherent nodes and their ground-truth labels. A Binary Cross Entropy loss $\mathcal{L}_{\text{Inc}}$ is for detecting incoherent regions. The detection loss $\mathcal{L}_{\text{Detect}}$ includes the localization and classification losses from the base detector, *e.g.* Faster R-CNN [35] or DETR detector. Finally, $\mathcal{L}_{\text{Coarse}}$ represents the loss for the initial coarse segmentation prediction used by [21]. $\lambda_{\{1,2,3,4\}}$ are hyper-parameter weights $\{1.0, 1.0, 1.0, 0.5\}$.

Implementation Details Mask Transfiner is implemented on both the two-stage detector Faster R-CNN [35] and query-based detector [4]. We design a 3-level quadtree and use the hyper-parameters and training schedules of Mask R-CNN implemented in Detectron2 [43] for the backbone and coarse mask head. The Mask Transfiner encoder consists of three standard transformer layers. Each layer has four attention heads with feature dimension at 256. In our ablation study, R-50-FPN [22] and Faster R-CNN with 1× learning schedule are adopted. For COCO leaderboard comparison, we adopt the scale-jitter with shorter image side randomly sampled from [640, 800], following training schedules in [26, 30]. More details are in the Supp. file.

4. Experiments

4.1. Experimental Setup

COCO We perform experiments on COCO dataset [33], where we train our networks on 2017*train* and evaluate our results on both the 2017*val* and 2017*test-dev*. We employ the standard AP metrics and the recently proposed boundary IoU metrics [10]. Notably, AP^B for boundary IoU is a measure focusing on boundary quality. Following [28], we also report $AP^\star$, which evaluates the *val* set of COCO with significantly higher-quality LVIS annotations [20] that can better reveal improvements in mask quality.

Cityscapes We report the results on Cityscapes [13], a high-quality instance segmentation dataset containing 2975, 500, 1525 images with resolution of 2048×1024 for training, validation and test respectively. Cityscapes focus on self-driving scenes with 8 categories (*e.g.*, car, person, bicycle).

BDD100K We further train and evaluate Mask Transfiner on the BDD100K [45] instance segmentation dataset, which has 8 categories with 120K high-quality instance mask annotations. We follow the standard practice, using 7k, 1k, 2k images for training, validation and testing respectively.

4.2. Ablation Experiments

We conduct detailed ablation studies on the COCO validation set, analyzing the impact of the proposed incoherent regions and individual components of Mask Transfiner.

Effect of the Incoherent Regions Table 1 presents an analysis on the properties of incoherent regions described in Section 3.1. It reveals they are critical to the final segmentation performance. Table 2 presents analyzes the effectiveness of the detected incoherent regions by replacing the refinement regions with full RoIs or detected object boundary regions. Due to memory limitation, the full RoIs only uses output size 28×28. The comparison shows the advantage of incoherent regions, with 1.8 AP and 0.7 AP gain over the use of full RoIs and detected boundary regions respectively.

To study the influence of incoherent regions on different pyramid levels, in Table 2, we also perform ablation experiments by removing the refinement regions of the Mask Transfiner in a level-wise order. We find that all three levels are beneficial to the final performance, while L_1 contributes most with 0.8 AP increase, where L_1 denotes the root level of Mask Transfiner with the smallest feature size.

Table 2. Effect of the incoherent regions on COCO *val* set. AP^B is evaluated Boundary IoU [10] while $AP^\star$ uses LVIS annotations.

Region Type	AP	AP^B	$AP^\star$	$AP^\star_{50}$
Full RoIs (28 × 28)	35.5	21.4	38.3	59.5
Boundary regions	36.6	23.8	40.1	60.2
Incoherent regions	**37.3**	**24.2**	**40.5**	**60.7**
Incoherent regions (**w/o** L_1)	36.5	23.5	39.8	59.7
Incoherent regions (**w/o** L_2)	36.8	23.8	40.2	60.1
Incoherent regions (**w/o** L_3)	36.7	23.6	40.0	59.9

Table 3. Effect of lower-level masks guidance in detecting incoherent regions on COCO *val*. AP and AP^B are final performance.

Lower-level Guidance	Acc	Recall	AP	AP^B
	79%	73%	36.6	23.7
✓	**84%**	**86%**	**37.3**	**24.2**

Table 4. Analysis of node encoding cues on COCO *val* set.

Fine	Coarse	Pos.	Context	AP	AP^B	$AP^\star$	$AP^\star_{50}$
✓				33.8	20.1	37.0	53.8
✓	✓			34.2	20.4	37.3	54.3
✓	✓	✓		36.8	23.9	40.1	60.1
✓	✓	✓	✓	**37.3**	**24.2**	**40.5**	**60.7**

Ablation on the Incoherent Regions Detector We evaluate the performance of the light-weight incoherent region detector by computing its recall and accuracy rates. In Table 3, with the guidance of the predicted incoherent mask up-sampled from lower level (Figure 5), the recall rate of detected incoherent regions has an obvious improvement from 74% to 86%, and the accuracy rate also increases from 79% to 84%. Note that recall rate is more important here to cover all the error-prone regions for further refinements.

Effect of Incoherent Points Encoding We analyze the effect of the four information cues in the incoherent points encoding. In Table 4, comparing to only using the fine-grained feature, the coarse segmentation features with semantic information brings a gain of 0.4 point AP. The positional encoding feature has a large influence on model performance by significantly improving 2.6 points on AP and 3.5 points on AP^B respectively. The positional encoding for incoherent points are crucial, because transformer architecture is permutation-invariant and the segmentation task is position-sensitive. The surrounding context feature further promotes the segmentation results from 36.8 AP to 37.3 AP by aggregating local neighboring details.

Influence of Quadtree Depths In Table 5, we study the influence on hierarchical refinement stages by constructing the quadtree in our Mask Transfiner with different depths. Depth 0 denotes the baseline using coarse head mask prediction **w/o** refinement steps. The output size grows twice larger than its preceding stage. By varying the output sizes from 28×28 to 224×224, the mask $AP^\star$ increases from 38.4 to 40.7 with increased tree depth. This reveals that models

Table 5. Analysis of the quadtree depth on the COCO *val* using R50-FPN as backbone.

Depth	Output size	AP	AP*	AP_L	AP_M	AP_S	FPS
0	28×28	35.2	37.6	50.3	37.7	17.2	12.3
1	28×28	35.5	38.4	50.9	38.1	17.2	10.6
2	56×56	36.2	39.1	51.9	38.7	17.3	8.9
3	112×112	**37.3**	40.5	52.9	**39.5**	**17.5**	7.1
4	224×224	37.1	**40.7**	**53.1**	39.3	17.4	5.2

Table 6. Mask Transfiner vs. MLP and CNN on COCO *val* set using ResNet-50-FPN.

Model	AP	AP^B	AP*	AP^*_{50}
CNN (full regions, 56 × 56)	35.7	21.8	38.7	58.8
MLP (full regions, 56 × 56)	36.1	23.4	39.2	59.2
MLP (PointRend [28], 112 × 112)	36.2	23.1	39.1	59.0
MLP (incoherent regions)	36.4	23.7	39.7	59.8
Mask Transfiner (D = 3, H = 4)	37.3	24.2	40.5	60.7
Mask Transfiner (D = 3, H = 8)	37.1	24.1	40.2	60.8
Mask Transfiner (D = 6, H = 4)	37.4	24.4	40.6	60.9

Table 7. Efficacy of Transfiner compared to standard attention models on COCO *val*. NLA denotes non-local attention [39].

Model	AP	FLOPs (G)	Memory (M)	FPS
NLA [39] (112×112)	36.3	24.6	8347	4.6
NLA [39] (224×224)	36.6	80.2	18091	2.4
Transformer [4] (28×28)	36.1	37.2	4368	6.9
Transformer [4] (56×56)	36.5	68.3	17359	2.1
Mask Transfiner (112×112)	**37.3**	**16.8**	**2316**	**7.1**
Mask Transfiner (224×224)	37.1	38.1	4871	5.2

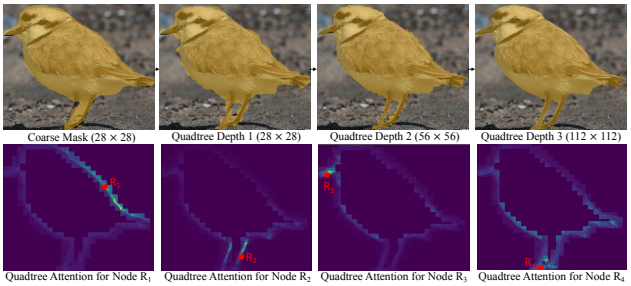


Figure 6. Qualitative results comparison between the coarse mask predictions by our baseline [22] and the refinement results with various depths of the quadtree built on detected incoherent regions. The bottom row visualizes the quadtree attention weights distribution in the sparse incoherent regions for four sampled red nodes.

with more levels and larger output sizes for an object indeed brings more gain to segmentation performance. The large objects benefit most from the increasing sizes with an improvement of 2.8 point in AP_L. We further find that the performance saturates when the output size is larger than 112×112, while the 3-stage Transfiner also has a lower computational cost and runs at 7.1 fps. Figure 6 visualizes results with increasing quadtree depths, where masks become substantially finer detail around object boundaries.

Mask Transfiner vs. MLP and CNN We compare different popular choices of the refinement networks, including the MLP and CNN structures. MLP is implemented with three hidden layers of 256 channels [28], while CNN is a FCN with four convolution layers with 3×3 kernels [21]. Note that for full refinement regions, CNN and MLP are limited to the RoI size 56 × 56 due to memory limitations, and CNN is not suitable for incoherent regions because uniform grids are required. In Table 6, our Mask Transfiner outperforms the MLP by 0.9 AP, benefiting from the non-local pixel-wise relation modeling, where we use the same incoherent regions on all three quadtree levels for fair comparison. Moreover, we investigate the influence of layer depth D and width W of Mask Transfiner and find that deeper and wider attention layers only lead to minor performance change. In Figure 6, we visualize the sparse quadtree attention maps of the last sequence encoder layer of the Transfiner, focusing on a few incoherent points. The encoder already seems to distinguish between foreground instances and background, where the neighboring attended regions of point R_1 are separated by the object boundary.

Efficacy of Quadtree Structure Table 7 compares Mask Transfiner with different attention mechanisms. Compared to pixels relation modeling using 3-layer non-local attention [39] or standard transformer [4, 37], Mask Transfiner not only obtains higher accuracy but also is very efficient in computation and memory consumption. For example, Mask Transfiner with multi-head attention uses 3 times less memory than the non-local attention given same output size, due to the small number of incoherent pixels. Compared to standard transformer operating on full RoI regions of much smaller size 56×56, the quadtree subdivision and inference allows Mask Transfiner to produce a high-resolution 224×224 prediction using only half of the FLOPs computation. Note that the standard transformer with output size 112×112 runs out of memory in our experiments.

Effect of Multi-level Joint Refinement Given incoherent nodes from the 3-level quadtree, Transfiner forms all of them into a sequence for joint refinement in single forward pass. In Table 8, we compare it with separately refining the quadtree nodes on each level with multiple sequences. The performance boost of 0.6 AP* shows the benefit of multi-scale feature fusion and richer context in global reasoning.

Effect of Quadtree Mask Propagation During inference, after Mask Transfiner has refined all incoherent points, we utilize a hierarchical coarse-to-fine mask propagation scheme along the quadtree levels to obtain the final predictions. Comparing to only correcting the labels of finest leaf nodes on the quadtree in Table 8, the propagation enlarges the refinement areas and improves the performance from 36.5 AP to 37.0 AP. The propagation brings negligible computation because the new labels for the quadrant leaf (coherent) nodes in intermediate tree levels are obtained via duplicating the refined label values of their parents.

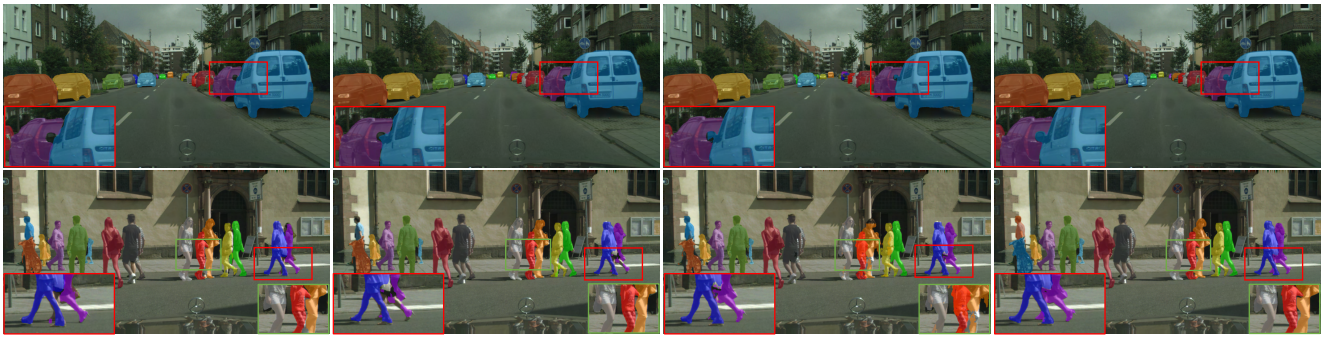
Table 8. Effect of the multi-level joint refinement (MJR) and quadtree mask propagation (QMP) on COCO *val* set.

MJR	QMP	AP	AP^B	AP*	AP^*_{50}
		36.5	23.7	39.6	59.7
✓		36.9	23.9	40.2	60.2
	✓	37.0	24.0	40.1	60.2
✓	✓	**37.3**	**24.2**	**40.5**	**60.7**

4.3. Comparison with State-of-the-art

We compare our approach with the state-of-the-art methods on the benchmarks COCO, Cityscapes and BDD100K, where Mask Transfiner outperforms all existing methods without bells and whistles, demonstrating efficacy on both two-stage and query-based segmentation frameworks.

COCO Table 9 compares Mask Transfiner with state-of-the-art instance segmentation methods on COCO dataset. Transfiner achieves consistent improvement on different



Mask R-CNN BMask R-CNN PointRend Mask Transfiner (Ours)

Figure 7. Qualitative comparisons with instance segmentation methods Mask R-CNN [21], BMask R-CNN [12], PointRend [28] and our Mask Transfiner on Cityscapes *val* set. Mask Transfiner produces more natural boundaries while revealing details for small parts, such as the rear mirrors of the car and the high-heeled shoes. Zoom in for better view. Refer to the supplemental file for more visual comparisons.

Table 9. Comparison with SOTA methods on COCO *test-dev* and *val* set. All methods are trained on COCO *train2017*. †: trained with DCN [49]. $AP^{\star}$ denotes evaluation using LVIS [20] annotation and AP^B denotes using Boundary IoU [10]. Type T denotes two-stage methods while Q denotes query-based methods.

Method	Backbone	Type	AP	$AP^{\star}_{val}$	AP^B_{val}	AP^{Box}	AP_S	AP_M	AP_L
Mask R-CNN [21]	R50-FPN	T	37.5	38.2	21.2	41.3	21.1	39.6	48.3
PointRend [28]	R50-FPN	T	38.1	39.7	23.5	41.5	18.8	40.2	49.4
B-MRCNN [12]	R50-FPN	T	37.8	39.8	23.5	41.6	19.7	40.3	49.6
BPR [36]	R50-FPN	T	38.4	40.2	24.3	41.3	20.2	40.5	49.7
Mask Transfiner	R50-FPN	T	39.4	42.3	26.0	41.8	22.3	41.2	50.2
Mask Transfiner†	R50-FPN	T	**40.5**	**43.1**	**26.8**	**43.2**	**22.8**	**42.3**	**52.5**
Mask R-CNN [21]	R101-FPN	T	38.8	39.3	23.1	43.1	21.8	41.4	50.5
PointRend [28]	R101-FPN	T	39.6	41.4	25.3	43.3	19.8	42.6	53.7
MS R-CNN† [24]	R101-FPN	T	39.6	41.1	25.0	44.1	18.9	42.7	55.1
HTC [6]	R101-FPN	T	39.7	42.5	25.4	**45.9**	21.0	42.2	53.5
RefineMask [47]	R101-FPN	T	39.4	42.3	26.8	43.8	21.6	42.0	53.1
BCNet [26]	R101-FPN	T	39.8	41.9	26.1	43.5	22.7	42.4	51.1
Mask Transfiner	R101-FPN	T	40.7	43.6	27.3	43.9	23.1	42.8	53.8
Mask Transfiner†	R101-FPN	T	**42.2**	**45.0**	**28.6**	45.8	**24.1**	**44.8**	**55.4**
ISTR [23]	R50-FPN	Q	38.6	39.5	23.0	46.8	22.1	40.4	50.6
QueryInst [17]	R50-FPN	Q	39.9	42.1	25.1	44.5	22.9	41.7	51.9
SOLQ [15]	R50-FPN	Q	39.7	39.8	23.3	**47.8**	21.5	42.5	53.1
Mask Transfiner	R50-FPN	Q	**41.6**	**45.4**	**28.2**	46.5	**24.2**	**44.6**	**55.2**

Table 10. Performance comparison between two-stage instance segmentation methods on Cityscapes *val* set using R50-FPN.

Method	AP^B	AP^B_{50}	AP	AP_{50}
Mask R-CNN (Baseline) [21]	11.4	37.4	33.8	61.5
PointRend [28]	16.7	47.2	35.9	61.8
BMask R-CNN [12]	15.7	46.2	36.2	62.6
Panoptic-DeepLab [9]	16.5	47.7	35.3	57.9
RefineMask [47]	17.4	49.2	37.6	63.3
Mask Transfiner (Ours)	**18.0**	**49.8**	**37.9**	**64.1**

Table 11. Performance comparison between instance segmentation methods on BDD100K *val* set.

Method	Backbone	AP_{mask}	AP_{box}
Mask R-CNN (Baseline) [21]	R101-FPN	20.5	26.1
Cascade Mask R-CNN [3]	R101-FPN	19.8	24.7
Mask R-CNN + DCNv2 [49]	R101-FPN	20.9	26.0
HRNet [38]	HRNet-w32	22.5	**28.2**
Mask Transfiner (Ours)	R101-FPN	**23.6**	26.2

backbones and object detectors, demonstrating its effectiveness by outperforming RefineMask [47] and BCNet [26] by 1.3 AP and 0.9 AP using R101-FPN and Faster R-CNN, and exceeding QueryInst [17] by 1.7 AP using query-based detector [4]. Note QueryInst consists of six-stage refinement in parallel with far more parameters to optimize. Besides, we find that Transfiner using Faster R-CNN and R50-FPN with much lower object detection performance still achieves comparable segmentation results with query-based methods [15,23] on mask AP, and over 2 points gain in boundary AP^B, further validating the higher AP achieved by Transfiner is indeed contributed by the fine-grained masks.

Cityscapes The results of Cityscapes benchmark is tabulated in Table 10, where Mask Transfiner achieves the best mask AP 37.6 and boundary AP^B 18.0. Our approach significantly surpasses existing SOTA methods, including PointRend [28] and BMask R-CNN [12] by a margin of 1.3 AP^B and 2.3 AP^B using the same Faster R-CNN detector. Compared to our baseline Mask R-CNN [21], Transfiner greatly improves the boundary AP from 11.4 to 18.0, which shows the effectiveness of the quadtree refinement.

BDD100K Table 11 shows results on BDD100K dataset, where Mask Transfiner obtains the highest AP_{mask} of 23.5 and outperforms the baseline [22] by 3 points under the comparable AP_{Box}. The significant advancements reveals the high accuracy of the predicted masks by Transfiner.

Qualitative Results Figure 7 shows qualitative comparisons on Cityscapes, where our Mask Transfiner produces masks with substantially higher precision and quality than previous methods [12, 21, 28], especially for the hard regions, such as the small rear mirrors and high-heeled shoes. Refer to supplementary file for more visual comparisons.

5. Conclusion

We present Mask Transfiner, a new high-quality and efficient instance segmentation method. Transfiner first detects and decomposes the image regions to build a hierarchical quadtree. Then, all points on the quadtree are transformed into to a query sequence for our transformer to predict final labels. In contrast to previous segmentation methods using convolutions limited by uniform image grids, Mask Transfiner produces high-quality masks with low computation and memory cost. We validate the efficacy of Transfiner on both the two-stage and query-based segmentation frameworks, and show that Transfiner achieves large performance advantages on COCO, Cityscapes and BDD100K. A current limitation is the fully supervised training required by our Mask Transfiner as well as the competing methods. Future work will strive towards relaxing this assumption.

References

[1] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: real-time instance segmentation. In *ICCV*, 2019. 2

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. 2

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. 2019. 8

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 2, 5, 6, 7, 8

[5] Hao Chen, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan. BlendMask: Top-down meets bottom-up for instance segmentation. In *CVPR*, 2020. 2

[6] 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. 2, 8

[7] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *CVPR*, 2018. 2

[8] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *ICCV*, 2019. 2

[9] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020. 8

[10] Bowen Cheng, Ross Girshick, Piotr Dollár, Alexander C Berg, and Alexander Kirillov. Boundary iou: Improving object-centric image segmentation evaluation. In *CVPR*, 2021. 6, 8

[11] Ho Kei Cheng, Jihoon Chung, Yu-Wing Tai, and Chi-Keung Tang. Cascadepsp: toward class-agnostic and very high-resolution segmentation via global and local refinement. In *CVPR*, 2020. 2

[12] Tianheng Cheng, Xinggang Wang, Lichao Huang, and Wenyu Liu. Boundary-preserving mask r-cnn. In *ECCV*, 2020. 1, 2, 8

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 6

[14] Henghui Ding, Xudong Jiang, Ai Qun Liu, Nadia Magnenat Thalmann, and Gang Wang. Boundary-aware feature propagation for scene segmentation. In *ICCV*, 2019. 2

[15] Bin Dong, Fangao Zeng, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Solq: Segmenting objects by learning queries. In *NeurIPS*, 2021. 1, 2, 3, 4, 8

[16] Qi Fan, Lei Ke, Wenjie Pei, Chi-Keung Tang, and Yu-Wing Tai. Commonality-parsing network across shape and appearance for partially supervised instance segmentation. In *ECCV*, 2020. 2

[17] Yuxin Fang, Shusheng Yang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Instances as queries. In *ICCV*, 2021. 1, 2, 8

[18] Raphael A Finkel and Jon Louis Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974. 1

[19] Ruohao Guo, Dantong Niu, Liao Qu, and Zhenbo Li. Sotr: Segmenting objects with transformers. In *ICCV*, 2021. 2

[20] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 6, 8

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 2, 3, 4, 5, 7, 8

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6, 7, 8

[23] Jie Hu, Liujuan Cao, Yao Lu, ShengChuan Zhang, Ke Li, Feiyue Huang, Ling Shao, and Rongrong Ji. Istr: End-to-end instance segmentation via transformers. *arXiv preprint arXiv:2105.00637*, 2021. 1, 2, 3, 8

[24] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *CVPR*, 2019. 1, 2, 8

[25] Lei Ke, Xia Li, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Prototypical cross-attention networks for multiple object tracking and segmentation. In *Advances in Neural Information Processing Systems*, 2021. 2

[26] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Deep occlusion-aware instance segmentation with overlapping bilayers. In *CVPR*, 2021. 2, 6, 8

[27] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Occlusion-aware video object inpainting. In *ICCV*, 2021. 2

[28] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 1, 2, 4, 6, 7, 8

[29] Weicheng Kuo, Anelia Angelova, Jitendra Malik, and Tsung-Yi Lin. Shapemask: Learning to segment novel objects by refining shape priors. In *ICCV*, 2019. 2

[30] Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *CVPR*, 2020. 2, 6

[31] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017. 2

[32] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 4

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 1, 2, 6

[34] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 1

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 2, 5, 6

[36] Chufeng Tang, Hang Chen, Xiao Li, Jianmin Li, Zhaoxiang Zhang, and Xiaolin Hu. Look closer to segment better: Boundary patch refinement for instance segmentation. 2021. 2, 8

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 7

[38] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2020. 1, 2, 8

[39] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 7

[40] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. *arXiv preprint arXiv:1912.04488*, 2019. 2

[41] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. In *NeurIPS*, 2020. 2

[42] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *CVPR*, 2021. 2

[43] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6

[44] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *CVPR*, 2020. 2

[45] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020. 6

[46] Yuhui Yuan, Jingyi Xie, Xilin Chen, and Jingdong Wang. Segfix: Model-agnostic boundary refinement for segmentation. In *ECCV*, 2020. 2

[47] Gang Zhang, Xin Lu, Jingru Tan, Jianmin Li, Zhaoxiang Zhang, Quanquan Li, and Xiaolin Hu. Refinemask: Towards high-quality instance segmentation with fine-grained features. In *CVPR*, 2021. 2, 8

[48] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-net: Towards unified image segmentation. In *NeurIPS*, 2021. 2

[49] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 8