SOTR: Segmenting Objects with Transformers

Ruohao Guo[*1,3], Dantong Niu[*2], Liao Qu[1,3], and Zhenbo Li[†1,3,4]

[1]College of Information and Electrical Engineering, China Agricultural University
[2]EECS department, University of California, Berkeley
[3]Key Laboratory of Agricultural Information Acquisition Technology, Ministry of Agriculture
[4]National Innovation Center for Digital Fishery, China Agricultural University

Abstract

Most recent transformer-based models show impressive performance on vision tasks, even better than Convolution Neural Networks (CNN). In this work, we present a novel, flexible, and effective transformer-based model for high-quality instance segmentation. The proposed method, Segmenting Objects with TRansformers (SOTR), simplifies the segmentation pipeline, building on an alternative CNN backbone appended with two parallel subtasks: (1) predicting per-instance category via transformer and (2) dynamically generating segmentation mask with the multi-level upsampling module. SOTR can effectively extract lower-level feature representations and capture long-range context dependencies by Feature Pyramid Network (FPN) and twin transformer, respectively. Meanwhile, compared with the original transformer, the proposed twin transformer is time- and resource-efficient since only a row and a column attention are involved to encode pixels. Moreover, SOTR is easy to be incorporated with various CNN backbones and transformer model variants to make considerable improvements for the segmentation accuracy and training convergence. Extensive experiments show that our SOTR performs well on the MS COCO dataset and surpasses state-of-the-art instance segmentation approaches. We hope our simple but strong framework could serve as a preferment baseline for instance-level recognition. Our code is available at https://github.com/easton-cau/SOTR.

1. Introduction

Instance segmentation, a fundamental task in computer vision, requires the correct prediction of each object instance and its per-pixel segmentation mask in an image.

---

*Equal contribution.
†Corresponding author. E-mail: lizb@cau.edu.cn

Figure 1: **Selected output of SOTR.** We combine CNN with transformer and obtain competitive qualitative results. Notice that not only are larger objects well delineated, targets with elaborate shapes can also get nice segmentation.

It becomes more challenging because of the contiguously increasing demands for precise separation of instances in complicated scenes with dense objects and accurate prediction of their masks at the pixel level. Modern instance segmentation approaches [24, 15] are typically built on CNN and follow the detect-then-segment paradigm, which consists of a detector used to identify and locate all objects, and a mask branch to generate segmentation masks. The success of this segmentation philosophy is attributed to the following favorable merits, i.e. translation equivariance and location, but faces the following obstacles: 1) CNN relatively lacks features' coherence in high-level visual semantic information to associate instances due to the limited receptive field, leading to the sub-optimal results on large objects; 2) Both the segmentation quality and inference speed rely heavily on the object detector, incurring inferior performance in complex scenarios.

To overcome these drawbacks, many recent studies tend to escape from the detect-then-segment manner toward bottom-up strategy [29, 31], which learns per-pixel embedding and instance-aware features, and then uses post-

processing techniques to successively group them into instances based on the embedding characteristics. Therefore, these methods can well retain position and local-coherence information. However, the main shortcomings of bottom-up models are unstable clustering (e.g., fragmented and joint masks) and poor generalization ability on the dataset with different scenes. Our SOTR (Figure 1 and 2) effectively learns position-sensitive features and dynamically generates instance masks following the basic principle of [37], without the post-processing grouping and the bound of bounding box's locations and scales.

Furthermore, inspired by the power of transformer in natural language processing (NLP) [4, 10, 35], dozens of works try to entirely substitute the convolution operation or combine the CNN-like architectures with transformers for feature extraction in vision tasks [11, 8, 5], which can easily capture global-range characteristics and naturally models long-distance semantic dependencies. In particular, self-attention, the key mechanism of transformers, broadly aggregates both feature and positional information from the whole input domain. Thus transformer-based models can better distinguish overlapped instances with the same semantic category, which makes them more suitable than CNN on high-level vision tasks. Nevertheless, insufficiencies still exist in these transformer-based approaches. On the one hand, the typical transformer does not behave well in extracting low-level features, leading to erroneous predictions on small objects. On the other hand, due to the extensive feature map, a large amount of memory and time are required, especially during the training stage.

To cope with these weaknesses, we propose an innovative bottom-up model called SOTR that ingeniously combines the advantages of CNN and transformer. More specifically, we adopt a new transformer model inspired by [20] to acquire global dependencies and extract high-level features for predictions in subsequent functional heads. Figure 2 shows the overall pipeline of our SOTR. It is composed of three parts, a CNN backbone, a transformer, and a multi-level upsampling module. An image is first fed to FPN to generate feature maps in multi-scale. After patch recombination and positional embedding, the transformer takes the clip-level feature sequences or blocks as inputs and further grasps the global-level semantic features as the powerful complement of the backbone. Then, part of the output feature is input to functional heads for the category and convolution kernel prediction. Finally, the multi-level upsampling module fuses the multi-scale features to a unified one to generate instance masks with the assistance of the dynamic convolution operation.

The focus of SOTR is to investigate ways to better utilize the semantic information extracted by the transformer. With the aim to reduce the memory and computational complexity of the conventional self-attention mechanism, we put forward twin attention, which adopts a sparse representation of the traditional attention matrix. We carry out a great deal of ablation experiments to explore the optimal architecture and hyper-parameters. In summary, not only does our SOTR provide a new framework for instance segmentation, but also it outperforms most of the CNN approaches on the MS COCO [27] dataset. Specifically, the overall contributions of our work are listed as follows:

- We introduce an innovative CNN-transformer-hybrid instance segmentation framework, termed SOTR. It can effectively model local connectivity and long-range dependencies leveraging CNN backbone and transformer encoder in the input domain to make them highly expressive. What's more, SOTR considerably streamlines the overall pipeline by directly segmenting object instances without relying on box detection.

- We devise the twin attention, a new position-sensitive self-attention mechanism, which is tailored for our transformer. This well-designed architecture enjoys a significant saving in computation and memory compared with original transformer, especially on large inputs for a dense prediction like instance segmentation.

- Apart from pure transformer based models, the proposed SOTR does not need to be pre-trained on large datasets to generalize inductive biases well. Thus, SOTR is easier applied to insufficient amounts of data.

- The performance of SOTR achieves 40.2% of AP with the ResNet-101-FPN backbone on the MS COCO benchmark, outperforming most of state-of-the-art approaches in accuracy. Furthermore, SOTR demonstrates significantly better performance on medium (59.0%) and large objects (73.0%), thanks to the extraction of global information by twin transformer.

2. Related work

2.1. Instance segmentation

Top-down instance segmentation. This approach deals with the problem by segmenting after detecting, which takes the inspiration of object detection tasks [33]. As the representative of anchor-based and two-stage methods, Mask R-CNN [15] added an extra branch for the instance segmentation in the proposed potential bounding boxes on the base of Faster R-CNN [33]. Also as one of the anchor-based methods, YOLACT [3] segmented instances in one-stage but two parallel subtasks: generating prototype masks and predicting mask coefficients for each instance. The final instance mask is the linear combination of the two. On the other hand, some works devoted to generating the segmentation masks within the anchor-free framework. Many of them derived from FCOS [34]. For example, CenterMask
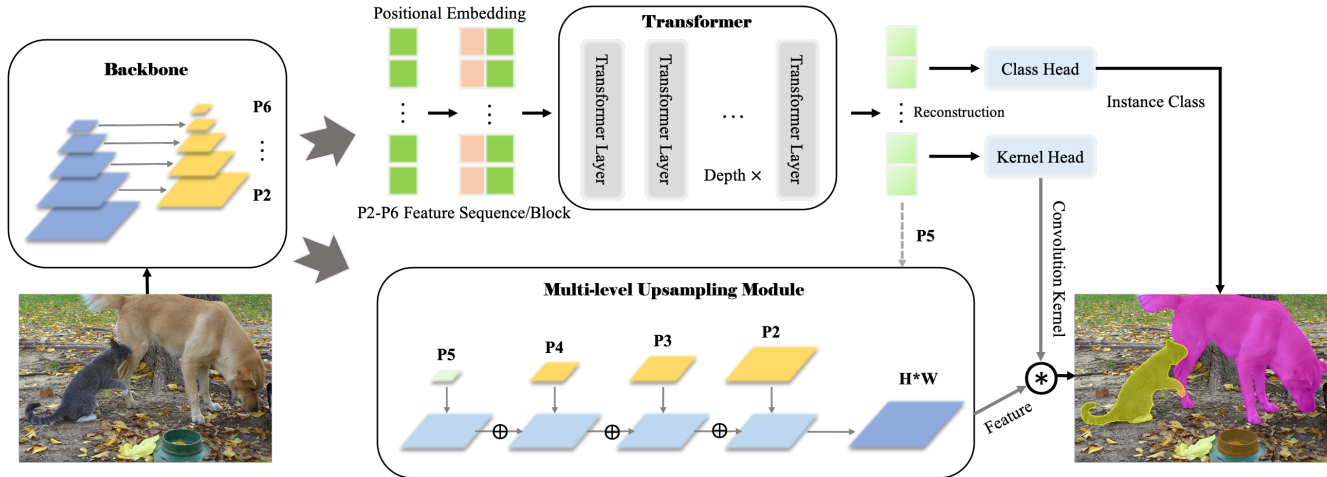
Figure 2: **Model overview.** SOTR builds on the simple FPN backbone with minimal modification. The model flattens FPN features P2-P6 and supplements them with positional embeddings before feeding them into the transformer model. Two heads are added after the transformer model to predict instance classes and produce dynamic convolution kernels. Multi-level upsampling module take P2-P4 features from FPN and P5 feature from transformer as inputs to generate final masks with dynamic convolution operation by ⊛.

[24] added a novel spacial attention-guided mask branch to FCOS to predict each detected box's segmentation mask.

Bottom-up instance segmentation. Different from top-down segmentation, this approach generates masks by clustering the pixels into each instance in an image. The typical methods include SSAP, SGN, etc. SGN [28] solved the problem by decomposing the clustering of instances with three sub-networks. In addition, the latest bottom-up method SOLO [36, 37] segmented the instance more directly. Instead of exploiting the relations between pixel pairwise, SOLO handled the clustering problem with classification. It categorized each grid and predicted the mask for each grid end-to-end without clustering. When the scenes are very complicated and dense objects exist in one image, lots of the computation and time will inevitably be lost on background pixels. However, our proposed SOTR takes an image as input, combines CNN with transformer module to extract the features, and directly makes predictions for class probabilities and instance masks.

2.2. Transformer in vision

Inspired by the great success of transformer in NLP, researchers propose to apply transformers to solve computer vision problems [14, 21, 18]. Following the standard transformer paradigms, Dosovitskiy et al. [11] presented a pure transformer model called Vision Transformer (ViT), which achieved state-of-the-art results on image classification tasks. To make the architecture of ViT as similar as the original transformer, the input image was reshaped into a sequence of flattened $2D$ patches and mapped to the corresponding embedding vectors with a trainable linear projection and position embeddings. The pure transformer model

could be naturally generalized to produce semantic segmentation by adding a FCN-based mask head. In Segmentation Transformer (SETR) [43], the framework built upon ViT with minimal modification and applied a progressive up-sampling strategy as the decoder to generate final masks.

While the above results are encouraging, transformer meets difficulties in extracting low-level features and lacks some inductive biases, so pure transformer models are overly dependent on pre-training on large datasets. These problems can be effectively solved by a combination with CNN backbone. Detection Transformer (DETR) [5] consisted of a standard CNN backbone and an encoder-decoder transformer for object detection. The former learned a $2D$ representation of the input image and generated the lower-resolution feature map. The latter predicted N objects (box coordinates and class labels) in parallel from the above flatten features with position information. However, two issues exist in DETR. Due to the feature mapping before the relation modeling in transformer, DETR not only suffers from the high computational cost but also performs poorly on small objects. Furthermore, DETR requires a longer training schedule to tune attention weights and focus on meaningful sparse locations.

For instance segmentation, DETR can be extended by appending a mask tower on top of the decoder outputs. Unlike these methods, we rethink the instance segmentation in a different manner and contribute a novel instance segmentation approach assembling CNN and transformer, called SOTR. The differences are apparent. First, SOTR follows standard FCN design and utilizes learnable convolutions to delineate each object region by location, directly segmenting instances in a box-free manner. Second, we adopt twin
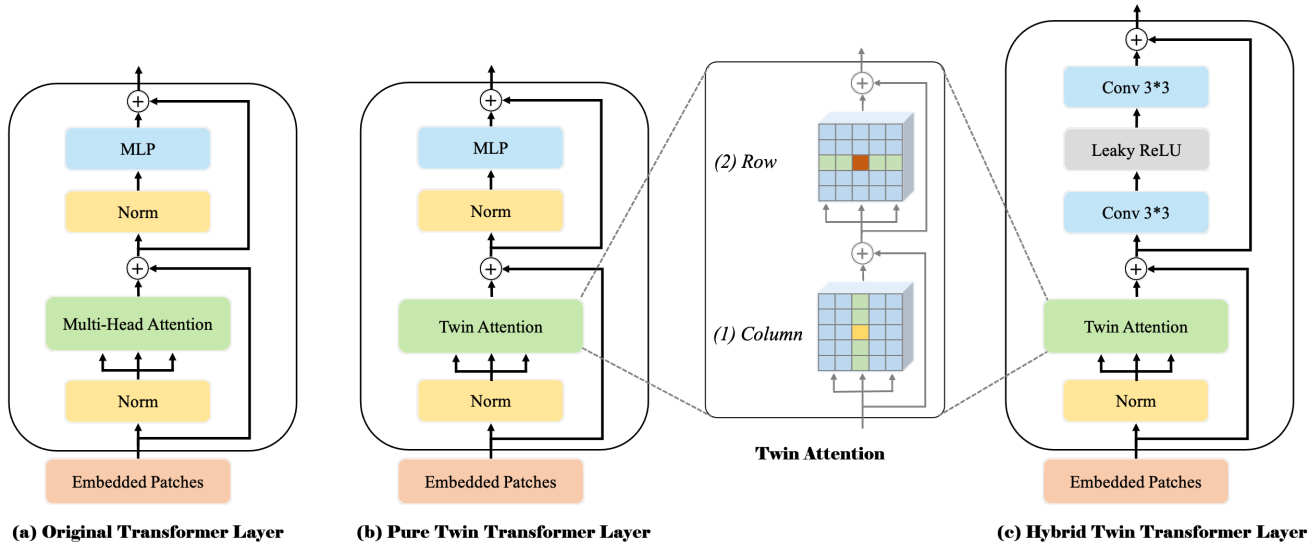
Figure 3: **Schematic illustration of three different transformer layer designs.** (a) The original transformer encoder from [35]. To better model long-range dependencies and improve computation efficiency, we introduce different transformer layer designs: (b) pure twin transformer layer and (c) hybrid twin transformer layer. Both layers are based on our designed twin attention that sequentially consists of (1) column- and (2) row-attention.

attention, an alternative self-attention autoregressive block, to significantly decrease computation and memory by decomposing the global spatial attention into independent vertical and horizontal attentions.

3. Methods

SOTR is a CNN-transformer hybrid instance segmentation model, which can simultaneously learn $2D$ representations and capture long-range information at ease. It follows a direct-segment paradigm, first dividing input feature maps into patches and then predicting per-patch class while dynamically segmenting each instance. Concretely, our model mainly consists of three parts: 1) a backbone to extract image features, especially lower-level and local features, from the input image, 2) a transformer to model global and semantic dependencies, which is appended with functional heads to predict per-patch class and convolution kernel respectively, and 3) a multi-level upsampling module to generate the final segmentation mask by performing dynamic convolution operation between the generating feature map and the corresponding convolution kernel. The overall framework is depicted in Figure 2.

3.1. Transformer

Twin attention. The self-attention mechanism is a key component of transformer models, which inherently captures full-image contexts and learns long-distance interactions between each element in the input sequence. However, self-attention has both quadratic time and memory complicity, incurring higher computational costs on high-

dimensional sequences such as images and hindering model scalability in different settings.

To mitigate the above-mentioned problems, we propose the twin attention mechanism to simplify the attention matrix with a sparse representation. Our strategy mainly limits the receptive field to a designed block pattern of fixed strides. It first computes the attention within each column while keeping elements in different columns independent. This strategy can aggregate contextual information between the elements on a horizontal scale (see Figure 3 (1)). Then, similar attention is performed within each row to fully exploit feature interactions across a vertical scale (shown in Figure 3 (2)). The attention in the two scales is sequentially connected to be the final one, which has a global receptive field and covers the information along the two dimensions.

Given feature maps $F_i \in \mathbb{R}^{H \times W \times C}$ at layer i of FPN, SOTR first splits the feature maps into $N * N$ patches $P_i \in \mathbb{R}^{N \times N \times C}$ and then stacks them into fixed blocks along with the vertical and horizontal directions. Position embeddings are added to the blocks to retain positional information, meaning that the position embedding spaces for the column and row are $1 * N * C$ and $N * 1 * C$. Both of the attention layers adopt the multi-head attention mechanism. To facilitate multi-layer connection and post-processing, all sub-layers in the twin attention produce $N \times N \times C$ outputs. Twin attention mechanism can effectively reduce the memory and computational complexity from standard $\mathcal{O}((H \times W)^2)$ to $\mathcal{O}(H \times W^2 + W \times H^2)$[1].

---
[1]The memory and computational complexity here is expressed in a more general form of H and W instead of N since the twin attention can process the input of any resolution, not limited to the square tensor.

Transformer layer. In this section, we introduce three different transformer layers based on encoder as our basic building block (as illustrated in Figure 3). The original transformer layer resembles the encoder used in NLP [35] (Figure 3 (a)), which comprises two parts: 1) a multi-head self-attention mechanism after a layer normalization [1], and 2) a multi-layer perception after a layer normalization. In addition, a residual connection [17] is employed to connect these two parts. Finally, a multi-dimensional sequence feature can be obtained as the output of the K serial connection of such transformer layers for subsequent predictions in different functional heads.

In order to make an optimal trade-off between the computational cost and the feature extracting effect, we follow the original transformer layer design and only substitute the multi-head attention with twin attention in the pure twin transformer layer (see Figure 3 (b)). In order to further boost the performance of twin transformer, we also design the hybrid twin transformer layer shown in Figure 3 (c). It adds two 3×3 convolution layers connected by a Leaky ReLU layer to each twin attention module. It is supposed the added convolution operation can be a useful complement to the attention mechanism, better capturing the local information and enhancing the feature representation.

Functional heads. The feature maps from transformer modules are input to different functional heads to make subsequent predictions. The class head includes a single linear layer to output a $N \times N \times M$ classification result, where M is the number of classes. Since each patch only assigns one category like YOLO [32] for one individual object whose center falls into the patch, we utilize multi-level prediction and share the heads across different feature levels to further improve the model performance and efficiency on objects of different scales. The kernel head is also composed of a linear layer, in parallel with the class head to output a $N \times N \times D$ tensor for subsequent mask generation, where the tensor denotes the $N \times N$ convolution kernels with D parameters. During training, Focal Loss [26] is applied to classification while all supervision for these convolution kernels comes from the final mask loss.

3.2. Mask

To construct mask feature representations for instance-aware and position-sensitive segmentation, a straightforward way is to make predictions on each feature map of different scales ([36, 12] and etc.). However, it will increase time and resources. Inspired by the Panoptic FPN [22], we design the multi-level upsampling module to merge the features from each FPN level and transformer to a unified mask feature. First, the relative low-resolution feature maps P5 with positional information are obtained from the transformer module and combined with P2-P4 in FPN to execute the fusion. For feature maps in each scale, several stages of

3×3 Conv, Group Norm [39] and ReLU are operated. Then P3-P5 are bilinear upsampled $2\times$, $4\times$, $8\times$, respectively to $(\frac{H}{4}, \frac{W}{4})$ resolution. Finally, after the processed P2-P5 are added together, a point-wise convolution and upsampling are executed to create final unified $H \times W$ feature maps.

For instance mask prediction, SOTR generates the mask for each patch by performing dynamic convolution operation on the above unified feature maps. Given predicted convolution kernels $K \in \mathbb{R}^{N \times N \times D}$ from the kernel head, each kernel is responsible for the mask generation of the instance in the corresponding patch. The detailed operation can be expressed as follows:

$$Z^{H \times W \times N^2} = F^{H \times W \times C} * K^{N \times N \times D} \qquad (1)$$

Where $*$ indicates the convolution operation, Z is the final generated mask with a dimension of $H \times W \times N^2$. It should be noted that the value of D depends on the shape of the convolution kernel, that is to say, D equals $\lambda^2 C$, where λ is kernel size. The final instance segmentation mask can be produced by Matrix NMS [37] and each mask is supervised independently by the Dice Loss [30].

4. Experiments

We conduct experiments on the challenging MS COCO dataset [27], which contains $123K$ images with 80-class instance labels. All models are trained on `train2017` subset and evaluated on `test-dev` subset. We also report the standard COCO metrics including average precision (AP), AP at IoU 0.5 (AP_{50}), 0.75 (AP_{75}) and AP for objects at different sizes AP_S, AP_M, and AP_L.

Implementation details. We train SOTR with SGD setting the initial learning rate of 0.01 with constant warm-up of $1k$ iterations and using weight decay of 10^{-4} and momentum of 0.9. For our ablation experiments, we train for $300K$ iterations with a learning rate drop by a factor of 10 at $210K$ and $250K$, respectively. Unless specified, all models are trained on 4 $V100$ GPUs of $32G$ RAM (take about 3-4 days), with batch size 8. The Python language is used for programming and the deep learning frameworks used are `PyTorch` and `Detectron2` [40].

4.1. Ablation experiments

We carry out a number of ablation experiments on the architectures and hyper-parameters to verify the effectiveness of parameter choice.

Table 1: **Backbone comparison results.** Better backbone brings expected gains: deeper neural network does better.

Backbone	AP	AP_S	AP_M	AP_L
Res-50-FPN	37.5	9.5	55.7	70.8
Res-101-FPN	**40.2**	**10.3**	**59.0**	**73.0**

Backbone architecture. We compare the performance of the different backbones on extracting features as shown in Table 1. We surprisingly find that SOTR with Res-50-FPN can already get 37.5% AP on the COCO and 70.8% AP on large objects. We note that our SOTR automatically benefits from deeper or advanced CNN backbones. In this ablation, results also show that the performance could be further improved by using a better backbone.

Table 2: **SOTR incorporating different transformers on COCO `test-dev`.** Note that all models are trained with same manners including $30K$ iterations, 8 batch size, etc. Under the same ResNet-101-FPN backbone, the hybrid transformer outperforms all other counterparts.

Transformer	AP	AP_S	AP_M	AP_L
Original	37.1	9.0	56.1	71.0
Pure Twin	39.7	9.9	**59.1**	**73.6**
Hybrid Twin	**40.2**	**10.3**	59.0	73.0

Transformer for feature encoding. We measure the performances of our model with three different transformers. Results of these variants are shown in Table 2. Our proposed pure and hybrid twin transformers surpass the original transformer by a large margin in all metrics, meaning that twin transformer architecture not only successfully captures long-range dependencies across vertical and horizontal dimensions but also is more suitable to be combined with CNN backbone to learn features and representations for images. For pure and twin transformers, the latter works much better. We assume the reason is that $3*3$ Conv can extract the local information and improve feature expression to enhance the rationality of the twin transformer.

Table 3: **Comparison of different depth.** Performance with Res-101-FPN backbone. Original and Twin denote the original transformer and the hybrid twin transformer.

Transformer	Depth	AP	Time(ms)	Memory
Original	6	36.2	147	6907M
	12	37.1	199	10696M
Twin	6	37.6	**113**	**3778M**
	12	**40.2**	161	5492M

Transformer depth. To verify the effect of transformer depth on SOTR, we conduct ablation experiments on the original transformer and the hybrid twin transformer, respectively. As shown in Table 3, both sets of experiments show that by increasing the depth of the transformer, we can increase AP while sacrificing inference time and memory. Also, the twin transformer brings 3.1% AP gains in contrast to the original transformer and reduces the memory footprint by about 50%, which shows the superiority

of this structure. However, as the transformer goes deeper, the attention maps gradually become similar, i.e., attention collapse. So transformer fails to learn more effective features and hinders model from getting expected performance gain. In further ablation experiment, we use hybrid twin transformer with `depth=12` for our baseline model if not specified.

Table 4: **Feature map substitution on multi-level upsampling process.** The check mark indicates whether to substitute P4 or P5 layer with feature maps generated by the 12-layer hybrid twin transformer.

P4	P5	AP	AP_S	AP_M	AP_L
		38.8	9.7	58.0	72.0
	✓	**40.2**	**10.3**	59.0	73.0
✓	✓	39.9	10.1	**59.1**	**73.7**

Multi-level Upsampling Module. In this ablation experiment, we explore the effect of feature maps generated by transformer on the multi-level upsampling module. As shown in Table 4, the model has the highest AP value when only replacing the P5 layer from FPN with the feature map generated by transformer. When replacing both the P4 and P5 layers, the model's AP value has a slight drop (-0.3%). This shows that using generated feature maps on more layers does not bring noticeable improvement in overall AP, and the P5 from transformer already enables predictions to be well position-sensitive. The reason why AP_M and AP_L are slightly improved 0.1% and 0.7% is that the P4 from transformer carries more global and larger object features than the P4 from FPN. In addition, SOLOv2 employs Coordconv on P5 to add positional information for segmentation, while SOTR substitutes it with transformers to gain such information and generate position-sensitive feature maps as shown in Figure 4.
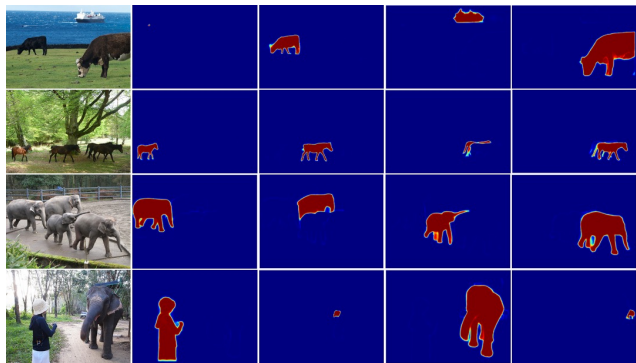

Figure 4: **SOTR behavior.** We show the visualization of the mask feature. For each row, the left side is the original picture, and the right side is the positional-sensitive mask correspond to it.

Figure 5: **Detailed comparison with other methods.** We compare the segmentation result of our method against Mask R-CNN [15], Blendmask [6] and SOLOv2 [37].The code and the trained model are provided by the original author. All models use ResNet-101-FPN as the backbone and are based on `Pytorch` and `Detectron2`. Our masks are of typically higher quality.

Table 5: **Dynamic convolution kernel vs. Static convolution kernel.** Learnable convolution kernel can considerably improve the results.

Twin	DCK	AP	AP_S	AP_M	AP_L
✓		38.6	9.5	57.7	72
	✓	39.7	**17.3**	42.9	57.4
✓	✓	**40.2**	10.3	**59.0**	**73.0**

Table 6: **Real-time setting comparison.** Metrics for models are obtained using their official trained models. The speed is reported on a single V100 GPU on COCO.

Model	Backbone	#param.	FLOPs	FPS	AP
YOLACT-550	R-50-FPN	140.23M	61.59G	**44.1**	28.2
PolarMask-600	R-50-FPN	**34.46M**	81.83G	21.7	27.6
SOTR-RT-736	R-50-FPN	38.20M	**60.31G**	25.2	**30.7**

Dynamic convolution. For mask generation, we have two options: to directly output instance masks in the static convolutional manner or to continually segment objects by dynamic convolution operation. The former one does not require the extra functional head to predict convolution kernels, while the latter one includes convolution kernel to generate final masks with the assistance of the fused feature. We compare these two modes in Table 5. As shown, SOTR without twin transformer achieves an AP of 39.7%, indicating the twin transformer brings 0.5% gains. In addition, the dynamic convolution strategy could improve the performance by almost 1.5% AP. We explain as follows. On the one hand, dynamic convolution significantly boosts the representation capability due to the non-linearity. On the other hand, dynamic convolution contribute to better and faster

convergence in training than its static counterpart.

Real-time model and comparison. SOTR focuses on boosting accuracy of instance segmentation while can be modified to be a real-time (RT) model with some sacrifice of the accuracy. The number of transformer layers of our designed SOTR-RT is reduced to two and the input shorter side is 736. Table 6 shows the performance of SOTR-RT models compared with others.

4.2. Main result

Quantitative results We compare SOTR to the state-of-the-art methods in instance segmentation on MS COCO `test-dev` in Table 7. SOTR with ResNet-101-FPN achieves a mask AP of 40.2%, which is much better than other modern instance segmentation methods. Com-

Table 7: **Quantitative results.** mask AP(%) on COCO `test-dev`. We compare our SOTR with state-of-the-art instance segmentation methods, and all entries in the table are *single-model* results. We denote the backbone architecture with `network-depth-feature`, where `Res` refer to ResNet [16]. Mask R-CNN* is the improved version with scale augmentation and longer training time($6\times$).

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FCIS [25]	Res-101-C5	29.5	51.5	30.2	8.0	31.0	49.7
MaskLab+ [7]	Res-101-C4	37.3	59.8	39.6	16.9	39.9	53.5
Mask R-CNN [15]	Res-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN*	Res-101-FPN	37.8	59.8	40.7	**20.5**	40.4	49.3
RetinaMask [13]	Res-101-FPN	34.7	55.4	36.9	14.3	36.7	50.5
MS R-CNN [19]	Res-101-FPN	38.3	58.8	41.5	17.8	40.4	54.4
TensorMask [9]	Res-101-FPN	37.1	59.3	39.4	17.4	39.1	51.6
ShapeMask [23]	Res-101-FPN	37.4	58.1	40.0	16.1	40.1	53.8
YOLACT [3]	Res-101-FPN	31.2	50.6	32.8	12.1	33.3	47.1
YOLACT++ [2]	Res-101-FPN	34.6	53.8	36.9	11.9	36.8	55.1
PolarMask [41]	Res-101-FPN	32.1	53.7	33.1	14.7	33.8	45.3
SOLO [36]	Res-101-FPN	37.8	59.5	40.4	16.4	40.6	54.2
BlendMask [6]	Res-101-FPN	38.4	60.7	41.3	18.2	41.5	53.3
CenterMask [38]	Hourglass-104	34.5	56.1	36.3	16.3	37.4	48.4
MEInst [42]	Res-101-FPN	33.9	56.2	35.4	19.8	36.1	42.3
SOLOv2 [37]	Res-101-FPN	39.7	60.7	42.9	17.3	42.9	57.4
SOTR	Res-101-FPN	**40.2**	**61.2**	**43.4**	10.3	**59.0**	**73.0**
SOLOv2 [37]	Res-DCN-101-FPN	41.7	63.2	45.1	**18.0**	45.0	61.6
SOTR	Res-DCN-101-FPN	**42.1**	**63.3**	**45.5**	11.5	**60.8**	**74.4**

pared with the traditional two-stage instance segmentation method Mask R-CNN [15], SOTR has achieved better prediction accuracy ($+2.4\%$ AP), and has achieved a considerable improvement in medium and large targets($+20.9\%$ in AP_M and $+20.6\%$ in AP_L). As one of the box-free algorithms, SOTR also has a significant improvement compared to SOLO [36] and PolarMask [41]. Furthermore, to the best of our knowledge, SOTR is the first method to achieves nearly 60% AP in medium objects and over 70% AP in medium and large objects.

Qualitative results We compare SOTR with the official Mask R-CNN [15], BlendMask [6] and SOLOv2 [37] models with ResNet-101 backbone [2]. The segmentation masks are displayed in Figure 5. Our SOTR performs better than Mask R-CNN and BlendMask in two cases: 1) objects with elaborate shapes easy to be omitted by other models (e.g. carrots in front of the train, recumbent elephants, drivers in the small cars), Mask R-CNN and BlendMask fail to detect them as positive instances. 2) objects overlapping with each other (e.g. person in front of the train), the two cannot separate them with accurate borders. SOTR is able to predict mask with sharper borders, while SOLOv2 tends to segment targets into separate parts (e.g. dividing the train into the head and the body) and sometimes cannot exclude the background from the image. Due to the introduction of

transformer, SOTR can better gain a comprehensive global-information to avoid such split on objects. Furthermore, compared with SOTR, SOLOv2 has a high false positive rate by assigning nonexistent objects as instance.

5. Conclusion

In this paper, we have proposed a new direct-segment instance segmentation method built upon CNN and transformer, which dynamically predicts the segmentation mask of each instance without object detectors, simplifying the overall pipeline. In order to process large inputs that are organized as multidimensional tensors, we design a transformer model variant based on the twin attention mechanism, which successfully reduces the memory and computational complexity to $\mathcal{O}(H \times W^2 + W \times H^2)$. In addition, it is efficient and easy to be integrated with different mainstream CNN backbones. Extensive ablation studies are conducted to verify the core factors of SOTR. Without bells and whistles, SOTR with ResNet-101-FPN backbone performs well and achieves 40.2% mAP on MS COCO dataset. We believe that our SOTR is capable of serving as a fundamental and strong baseline for instance segmentation tasks.

6. Acknowledgments

---

[2]To make a fair comparison with Mask R-CNN, BlendMask, and SOLOv2, the code base we use is `Detectron2`. Recently released `Detectron2` originates from `maskrcnn_benchmark` with significant enhancements for performance.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Y. Lee. Yolact++: Better real-time instance segmentation. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2020.

[3] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact: Real-time instance segmentation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9156–9165, 2019.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing.

[6] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan. Blendmask: Top-down meets bottom-up for instance segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8570–8578, 2020.

[7] Liang-Chieh Chen, A. Hermans, G. Papandreou, Florian Schroff, P. Wang, and H. Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4013–4022, 2018.

[8] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

[9] Xinlei Chen, Ross B. Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2061–2069, 2019.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[12] Ali Farhadi and Joseph Redmon. Yolov3: An incremental improvement. *Computer Vision and Pattern Recognition, cite as*, 2018.

[13] Cheng-Yang Fu, M. Shvets, and A. Berg. Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free. *ArXiv*, abs/1901.03353, 2019.

[14] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer. *arXiv preprint arXiv:2012.12556*, 2020.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[18] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[19] Z. Huang, Lichao Huang, Yongchao Gong, C. Huang, and Xinggang Wang. Mask scoring r-cnn. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6402–6411, 2019.

[20] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 603–612, 2019.

[21] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021.

[22] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[23] Weicheng Kuo, A. Angelova, Jitendra Malik, and Tsung-Yi Lin. Shapemask: Learning to segment novel objects by refining shape priors. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9206–9215, 2019.

[24] Y. Lee and J. Park. Centermask: Real-time anchor-free instance segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13903–13912, 2020.

[25] Y. Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. *2017*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4438–4446, 2017.

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[27] Tsung-Yi Lin, M. Maire, Serge J. Belongie, James Hays, P. Perona, D. Ramanan, Piotr Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[28] S. Liu, J. Jia, S. Fidler, and R. Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3516–3524, 2017.

[29] Yiding Liu, Siyu Yang, B. Li, W. Zhou, Jizheng Xu, H. Li, and Y. Lu. Affinity derivation and graph merge for instance segmentation. In *ECCV*, 2018.

[30] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016.

[31] D. Neven, Bert De Brabandere, M. Proesmans, and L. Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8829–8837, 2019.

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[33] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[34] Z. Tian, C. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9626–9635, 2019.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[36] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. SOLO: Segmenting objects by locations. In *Proc. Eur. Conf. Computer Vision (ECCV)*, 2020.

[37] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[38] Yuqing Wang, Zhaoliang Xu, H. Shen, B. Cheng, and Lirong Yang. Centermask: Single shot instance segmentation with point representation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9310–9318, 2020.

[39] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

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

[41] 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. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12190–12199, 2020.

[42] Rufeng Zhang, Zeyong Tian, Chunhua Shen, Mingyu You, and Y. Yan. Mask encoding for single shot instance segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10223–10232, 2020.

[43] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *arXiv preprint arXiv:2012.15840*, 2020.