Supplementary Material for Rethinking Transformer-based Set Prediction for Object Detection

Anonymous ICCV submission

Paper ID 5484

A. Preliminaries

A.1. Transformer and Detection Transformer

As this work aims to improve the DEtection TRansformer (DETR) model [3], for completeness, we describe its architecture in more details.

Encoder-decoder framework DETR can be formulated in an encoder-decoder framework [5]. The encoder of DETR takes the features processed by the CNN backbone as inputs and generates the context representation, and the non-autoregressive decoder of DETR takes the object queries as inputs and generates the detection results conditional on the context.

Multi-head attention Two types of multi-head attentions are used in DETR: multi-head self-attention and multi-head cross-attention. A general attention mechanism can be formulated as the weighted sum of the value vectors V using query vectors Q and key vectors K:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{\text{model}}}}\right) \cdot V, \qquad (1)$$

where d_{model} represents the dimension of hidden representations. For self-attention, Q, K, and V are hidden representations of the previous layer. For cross-attention, Q refers to hidden representations of the previous layer, whereas K and V are context vectors from the encoder. Multi-head variant of the attention mechanism allows the model to jointly attend to information from different representation subspaces, and is defined as:

$$\text{Multi-head}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)W^O,$$

$$\text{head}_h = \text{Attention}((P_Q + Q)W_h^Q, (P_K + K)W_h^K, VW_h^V),$$

where $W_h^Q, W_h^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_h^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $W_h^O \in \mathbb{R}^{Hd_v \times d_{\text{model}}}$ are projection matrices, H is the number of attention heads, d_k and d_v are the hidden sizes of queries/keys and values per head, and P_Q and P_K are positional encoding.

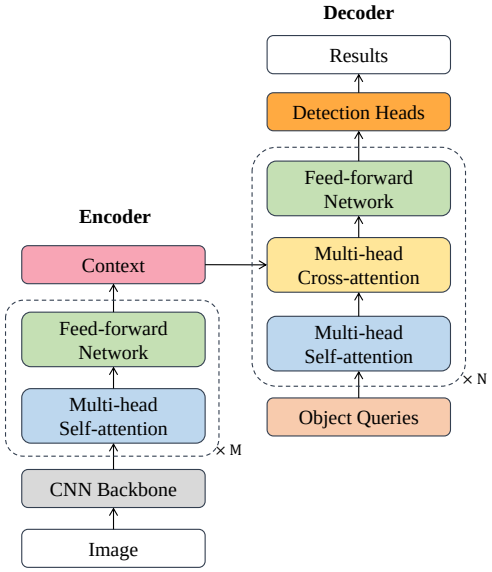

Figure 1. A detailed illustration of the DETR architecture. Residual connection and layer normalization are omitted.

Feed-forward network The position-wise Feed-Forward Network (FFN) is applied after multi-head attentions in both encoder and decoder. It consists of a two-layer linear transformation with ReLU activation:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2, \qquad (2)$$

where $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{FFN}}}$, $W_2 \in \mathbb{R}^{d_{\text{FFN}} \times d_{\text{model}}}$, $b_1 \in \mathbb{R}^{d_{\text{FFN}}}$, $b_2 \in \mathbb{R}^{d_{\text{model}}}$, and d_{FFN} represents the hidden size of FFN.

Stacking Multi-head attention and feed-forward network are stacked alternately to form the encoder and the decoder, with residual connections [9] and layer normalization [1]. Figure 1 shows a detailed illustration of the DETR architecture.

A.2. Faster R-CNN

Faster R-CNN [17] is a two-stage object detection model, developed based on previous work of R-CNN [7]

and Fast R-CNN [6]. With Region Proposal Networks (RPN), Faster R-CNN significantly improves the accuracy and efficiency of two-stage object detection.

Region Proposal Networks The first module of Faster R-CNN is a deep fully convolutional network, named the Region Proposal Network (RPN) that proposes Regions of Interest (RoIs). RPN takes the feature maps of a image as input, and outputs a set of rectangular object proposals with their objectness scores. RPN contains a shared 3×3 convolutional layer and two sibling 1×1 convolutional layers for regression and classification respectively. At each sliding-window location, RPN produces k proposals. The proposals are parameterized relative to k reference boxes called anchors. In Fast R-CNN, 3 scales and 3 aspect ratios of anchors are used, so there are $k = 9$ anchors for each sliding window. For each anchor, the regression head outputs 4 coordinate parameters $\{t_x, t_y, t_w, t_h\}$ that encode the location and size of bounding boxes, and the classification head outputs 2 scores $\{p_{\text{pos}}, p_{\text{neg}}\}$ that estimate probability of existence of object in the box.

Fast R-CNN The second part is the Fast R-CNN detector that uses each proposal from RPN to refine the detection. To reduce redundancy, non-maximum suppression (NMS) is applied on the proposals, and only the top ranked proposals can be used by Fast R-CNN. Then, RoI Pooling or RoI Align [8] is used to extract features from the backbone feature map at the given proposal regions, such that the input to the Fast R-CNN detector has fixed spatial size for each proposal. At this stage, Fast R-CNN outputs bounding box regression parameters and classification scores to refine the region proposals. Again, NMS is required to reduce duplication in the detection results.

A.3. FCOS

Fully Convolutional One-Stage Object Detection (FCOS) [20] is a recent anchor-free, per-pixel detection framework that has achieved state-of-the-art one-stage object detection performance.

Per-Pixel Prediction In contrast to anchor-based object detectors, FCOS formulates the task in a per-pixel prediction fashion, that is, the target bounding boxes are regressed at each location on the feature map, without referencing predefined anchors. A location on the feature map is considered as a positive sample if its corresponding position on the input image falls into any ground-truth box. If one location falls into the overlap of multiple ground-truth boxes, the smallest one is selected. Experiments show that with multi-level prediction and FPN [13], this ambiguity does not affect the overall performance.

Network Outputs In FCOS, there are two branches after the feature maps from the backbone. The first branch has 4 convolutional layers and two sibling layers that outputs C classification scores and a "center-ness" score. The center-ness depicts the normalized distance from the location to the center of the object that the location is responsible for. The center-ness ranges in $[0, 1]$ and is trained with binary cross entropy loss. During test, the center-ness is multiplied to the classification score, thus the possibly low-quality bounding boxes that are far away from the center of objects will have less weight in NMS. The second branch has 4 convolutional layers and a bounding box regression layer that outputs the distance from the location to the four sides of the box. The prediction head is shared across multiple feature levels.

B. Detailed Experimental Settings

We provide more details about the default settings of our implementation.

Backbone We use ResNet-50 and ResNet-101 [9] as the backbone, and a Feature Pyramid Network [13] is built on the $\{C_3, C_4, C_5\}$ feature maps from ResNet to produce feature pyramid $\{P_3, P_4, P_5, P_6, P_7\}$. If specified with DCN, we use Deformable ConvNets v2 [27] in the last three stages of ResNet. The feature maps have 256 channels.

Data augmentation We follow the default setting of Detectron2 [22] for data augmentation. Specifically, we use scale augmentation to resize the input images such that the shortest side is in $\{640, 672, 704, 736, 768, 800\}$, and the longest is no larger than 1333. Besides scale augmentation, we also randomly flip training images horizontally.

Loss We use our proposed faster set prediction training loss for classification, and a combination of L1 and Generalized IoU [18] losses for regression. Focal loss [14] is used for weighting positive and negative examples in classification for both TSP-FCOS and TSP-RCNN. Unlike DETR [3], we do not apply auxiliary losses after each encoder layer. We find this end-to-end scheme improves the model performance.

Optimization We use AdamW [15] to optimize the Transformer component, and SGD with momentum 0.9 to optimizer the other parts in our detector. For the 36-epoch ($3\times$) schedule, we train the detector for 2.7×10^5 iterations with batch size 16. The learning rate is set to 10^{-4} for AdamW, and 10^{-2} for SGD in the beginning, and both multiplied by 0.1 at 1.8×10^5 and 2.4×10^5 iterations. We also use linear learning rate warm-up in the first 1000 iterations. The weight decay is set to 10^{-4}. We apply gradient clipping

Model	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN [17]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2
Fitness NMS [21]	ResNet-101	41.8	60.9	44.9	21.5	45.0	57.5
Libra RCNN [16]	ResNet-101	41.1	62.1	44.7	23.4	43.7	52.5
Cascade RCNN [2]	ResNet-101	42.8	62.1	46.3	23.7	45.5	55.2
TridentNet [12]	ResNet-101-DCN	46.8	67.6	51.5	28.0	51.2	60.5
TSD [19]	ResNet-101	43.2	64.0	46.9	24.0	46.3	55.8
Dynamic RCNN [24]	ResNet-101	44.7	63.6	49.1	26.0	47.4	57.2
Dynamic RCNN [24]	ResNet-101-DCN	46.9	65.9	51.3	28.1	49.6	60.0
RetinaNet [14]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
FSAF [26]	ResNet-101	40.9	61.5	44.0	24.0	44.2	51.3
FCOS [20]	ResNet-101	41.5	60.7	45.0	24.4	44.8	51.6
MAL [10]	ResNet-101	43.6	62.8	47.1	25.0	46.9	55.8
RepPoints [23]	ResNet-101-DCN	45.0	66.1	49.0	26.6	48.6	57.5
ATSS [25]	ResNet-101	43.6	62.1	47.4	26.1	47.0	53.6
ATSS [25]	ResNet-101-DCN	46.3	64.7	50.4	**27.7**	**49.8**	58.4
TSP-FCOS	ResNet-101	46.1	65.8	50.3	27.3	49.0	58.2
TSP-FCOS	ResNet-101-DCN	**46.8**	**66.4**	**51.0**	27.6	49.5	**59.0**

Table 1. Compare TSP-FCOS with state-of-the-art models on COCO 2017 test set (single-model and single-scale results). Underlined and **bold** numbers represent the best one-stage model with ResNet-101 and ResNet-101-DCN backbone, respectively.

Model	AP	AP_S	AP_M	AP_L	FLOPs	#Params
FCOS	41.0	26.2	44.6	52.2	177G	36.4M
FCOS-larger	41.5	26.0	45.2	52.3	199G	37.6M
TSP-FCOS	**43.1**	**26.6**	**46.8**	**55.9**	189G	51.5M
Faster RCNN	40.2	24.2	43.5	52.0	180G	41.7M
Faster RCNN-larger	40.9	24.4	44.1	54.1	200G	65.3M
TSP-RCNN	**43.8**	**28.6**	**46.9**	**55.7**	188G	63.6M

Table 2. Evaluation results on COCO 2017 validation set of models under various FLOPs. ResNet-50 is used as backbone.

for the Transformer part, with a maximal L_2 gradient norm of 0.1.

Longer training schedule We also use a 96-epoch (8×) schedule in the paper. The 96-epoch (8×) schedule will resume from 36-epoch (3×) schedule's model checkpoint in the 24^{th} epoch (i.e., 1.8×10^5 iterations), and continue training for 72 epoch (i.e., 5.4×10^5 iterations). The learning rate is multiplied by 0.1 at 4.8×10^5 and 6.4×10^5 iterations. In the 8× schedule, we will further apply random crop augmentation. We follow the augmentation strategy in DETR [3], where a train image is cropped with probability 0.5 to a random rectangular patch which is then resized again to 800-1333.

C. More Details of Encoder-only DETR

Our encoder-only DETR is also trained with the Hungarian loss for set prediction, but the bounding box regression process is a bit different. In original DETR, bounding box regression is reference-free, where DETR directly predicts the normalized center coordinates $(cx, cy) \in [0, 1]^2$, height and width $(w, h) \in [0, 1]^2$ of the box w.r.t. the input image. In encoder-only DETR, as each prediction is based on a feature point of Transformer encoder output, we will use the feature point coordinates (x_r, y_r) as the reference point of regression:

$$cx = \sigma(b_1 + \sigma^{-1}(x_r)), cy = \sigma(b_2 + \sigma^{-1}(y_r))$$

where $\{b_1, b_2\}$ are from the output of regression prediction.

Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Deformable DETR	43.8	62.6	47.7	26.4	47.1	58.0
+ iterative refinement	45.4	64.7	49.0	26.8	48.3	61.7
++ two-stage*	**46.2**	**65.2**	**50.0**	28.8	**49.2**	**61.7**
TSP-RCNN	44.4	63.7	49.0	29.0	47.0	56.7
+ iterative refinement	45.4	63.1	49.6	**29.5**	48.5	58.7

Table 3. Evaluation results on COCO 2017 validation set of TSP-RCNN and Deformable DETR with iterative refinement. All models are trained with 50 epochs and a batch size of 32. * Please refer to the original Deformable DETR paper for the definition of two-stage Deformable DETR.

D. Comparison between TSP-RCNN and Deformable DETR with Iterative Refinement

Inspired by Deformable DETR [28], we conduct experiments of TSP-RCNN which also iteratively refines the prediction boxes in a cascade style [2]. Here we implement a simple two-cascade scheme, whether the dimension of fully connected detection head and Transformer feed-forward network are reduced from 12544-1024-1024 and 512-2048-512 to 12544-512 and 512-1024-512, respectively, to maintain a similar number of parameters and FLOPs as the orig-

inal model. To make a fair comparison, we also follow the experimental setting of Deformable DETR where a 50-epoch training schedule with batch size 32 is used.

Table 3 shows the results of TSP-RCNN and Deformable DETR with iterative refinement. From the results, we can see that without iterative refinement, TSP-RCNN outperforms Deformable DETR with the same training setting. The iterative refinement process can improve the performance of TSP-RCNN by 1 AP point. We can also find that both with iterative refinement, TSP-RCNN slightly underperforms Deformable DETR. We believe this is because Deformable DETR utilizes $D = 6$ decoder refinement iterations, while we only conduct experiments with two refinement iterations. How to efficiently incorporate multiple refinement iterations into the TSP-RCNN model is left as future work.

E. Comparison under similar FLOPs

Compared to original FCOS and Faster RCNN, our TSP-FCOS and TSP-RCNN use an additional Transformer encoder module. Therefore, it is natural to ask whether the improvements come from more computation and parameters. Table 2 answers this question by applying stronger baseline models to the baseline models. For Faster RCNN, we first apply two unshared convolutional layers to P_3-P_7 as a stronger RPN, and then change the original 12544-1024-1024 fully-connected (fc) detection head to 12544-2048-2048-2048. This results in a Faster RCNN model with roughly 200 GFLOPs and 65.3M parameters. For FCOS, we evaluate a FCOS model with roughly 199 GFLOPs, where we add one more convolutional layer in both classification and regression heads. From Table 2, we can see that while adding more computation and parameters to baselines can slightly improve their performance, such improvements are not as significant as our TSP mechanism.

F. Compare TSP-FCOS with State-of-the-Arts

For completeness, we also compare our proposed TSP-FCOS model with other state-of-the-art detection models [17, 21, 2, 19, 14, 26, 20, 4, 11, 23, 25] that also use ResNet-101 backbone or its deformable convolution network (DCN) [27] variant in Table 1. A $8\times$ schedule and random crop augmentation is used. The performance metrics are evaluated on COCO 2017 test set using single-model and single-scale detection results. We can see that TSP-FCOS achieves state-of-the-art performance among one-stage detectors in terms of the AP score. But comparing Table 4 in the main paper and Table 1, we can also find that TSP-FCOS slightly under-performs our proposed TSP-RCNN model.

References

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

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 3, 4

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020. 1, 2, 3

[4] Yuntao Chen, Chenxia Han, Naiyan Wang, and Zhaoxiang Zhang. Revisiting feature alignment for one-stage object detection. *arXiv preprint arXiv:1908.01570*, 2019. 4

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 1

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[8] 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. 2

[9] 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. 1, 2

[10] Wei Ke, Tianliang Zhang, Zeyi Huang, Qixiang Ye, Jianzhuang Liu, and Dong Huang. Multiple anchor learning for visual object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10206–10215, 2020. 3

[11] Tao Kong, Fuchun Sun, Huaping Liu, Yuning Jiang, Lei Li, and Jianbo Shi. Foveabox: Beyound anchor-based object detection. *IEEE Transactions on Image Processing*, 29:7389–7398, 2020. 4

[12] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 6054–6063, 2019. 3

[13] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2

[14] 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. 2, 3, 4

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[16] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 821–830, 2019. 3

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 3, 4

[18] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019. 2

[19] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11563–11572, 2020. 3, 4

[20] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 9627–9636, 2019. 2, 3, 4

[21] Lachlan Tychsen-Smith and Lars Petersson. Improving object localization with fitness nms and bounded iou loss. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6877–6885, 2018. 3, 4

[22] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019. 2

[23] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9657–9666, 2019. 3, 4

[24] Hongkai Zhang, Hong Chang, Bingpeng Ma, Naiyan Wang, and Xilin Chen. Dynamic r-cnn: Towards high quality object detection via dynamic training. *arXiv preprint arXiv:2004.06002*, 2020. 3

[25] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9759–9768, 2020. 3, 4

[26] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 840–849, 2019. 3, 4

[27] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 2, 4

[28] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 3