Rethinking Transformer-based Set Prediction for Object Detection

Zhiqing Sun*   Shengcao Cao*   Yiming Yang   Kris Kitani
Carnegie Mellon University

{zhiqings,shengcao,yiming,kkitani}@cs.cmu.edu

Abstract

DETR is a recently proposed Transformer-based method which views object detection as a set prediction problem and achieves state-of-the-art performance but demands extra-long training time to converge. In this paper, we investigate the causes of the optimization difficulty in the training of DETR. Our examinations reveal several factors contributing to the slow convergence of DETR, primarily the issues with the Hungarian loss and the Transformer cross-attention mechanism. To overcome these issues we propose two solutions, namely, TSP-FCOS (Transformer-based Set Prediction with FCOS) and TSP-RCNN (Transformer-based Set Prediction with RCNN). Experimental results show that the proposed methods not only converge much faster than the original DETR, but also significantly outperform DETR and other baselines in terms of detection accuracy.

1. Introduction

Object detection aims at finding all objects of interest in an image and predicting their category labels and bounding boxes, which is essentially a set prediction problem, as the ordering of the predicted objects is not required. Most of the state-of-the-art neural detectors [24, 28, 22, 29, 42, 30, 13] are developed in a detect-and-merge fashion that is, instead of directly optimizing the predicted set in an end-to-end fashion, those methods usually first make predictions on a set of region proposals or sliding windows, and then perform a post-processing step (e.g., "non-maximum suppression" or NMS) for merging the the detection results in different proposals or windows that might belong to the same object. As the detection model is trained agnostically with respect to the merging step, the model optimization in those object detectors is not end-to-end and arguably sub-optimal.

DEtection TRansformer (DETR) [3] is recently proposed as the first fully end-to-end object detector. It uses Transformer [36] to directly output a final set of predictions without further post-processing. However, it takes extra-long training time to converge. For example, the popular Faster RCNN model [30] only requires about 30 epochs to convergence, but DETR needs 500 epochs, which takes at least 10 days on 8 V100 GPUs. Such expensive training cost would be practically prohibitive in large applications. Therefore, in what manner should we accelerate the training process towards fast convergence for DETR-like Transformer-based detectors is a challenging research question and is the main focus of this paper.

For analyzing the causes of DETR's optimization difficulty we conduct extensive experiments and find that the cross-attention module, by which the Transformer decoder obtains object information from images, is mainly responsible for the slow convergence. In pursuit of faster convergence, we further examine an encoder-only version of DETR by removing the cross-attention module. We find that the encoder-only DETR yields a substantial improvement for the detection of small objects in particular but sub-optimal performance on large objects. In addition, our analysis shows that the instability of the bipartite matching in DETR's Hungarian loss also contributes to the slow convergence.

Based on the above analysis we propose two models for significantly accelerating the training process of Transformer-based set prediction methods, both of which can be regarded as improved versions of encoder-only DETR with feature pyramids [21]. Specifically, we present TSP-FCOS (Transformer-based Set Prediction with FCOS) and TSP-RCNN (Transformer-based Set Prediction with RCNN), which are inspired by a classic one-stage detector FCOS [34] (Fully Convolutional One-Stage object detector) and a classic two-stage detector Faster RCNN [30], respectively. A novel Feature of Interest (FoI) selection mechanism is developed in TSP-FCOS to help Transformer encoder handle multi-level features. To resolve the instability of the bipartite matching in the Hungarian loss, we also design a new bipartite matching scheme for each of our two models for accelerating the convergence in training. In our evaluation on the COCO 2017 detection benchmark [23] the proposed methods not only converge much faster than

---

*indicates equal contribution.

the original DETR, but also significantly outperform DETR and other baselines in terms of detection accuracy.

2. Background

2.1. One-stage and Two-stage Object Detectors

Most modern object detection methods can be divided into two categories: One-stage detectors and two-stage detectors. Typical one-stage detectors [24, 28, 22] directly make predictions based on the extracted feature maps and (variable-sized) sliding-window locations in a image, while two-stage detectors [30, 13] first generate region proposals based on sliding-window locations and then refine the detection for each proposed region afterwards. In general, two-stage detectors are more accurate but also computationally more expensive than one-stage detectors. Nevertheless, both kinds of detectors are developed in a detection-and-merge fashion, *i.e.*, they require a post-processing step to ensure that each detected object has only one region instead of multiple overlapping regions as detection results. In other words, many state-of-the-art object detection methods do not have an end-to-end training objective with respect to set prediction. For example, these methods [11, 30, 28] usually assign object labels to sliding windows or region proposals based on Intersection-over-Unit (IoU) scores, which may yield multiple regions being assigned to the same object. Thus, a post-processing step with Non-Maximum Suppression (NMS) is required for these methods to remove the detection duplicates.

2.2. DETR with an End-to-end Objective

Different from the aforementioned popular object detectors, DEtection TRansformer (DETR) [3] presents the first method with an end-to-end optimization objective for set prediction. Specifically, it formulates the loss function via a bipartite matching mechanism. Let us denote by $y = \{y_i\}_{u=1}^{M}$ the ground truth set of objects, and $\hat{y} = \{\hat{y}_i\}_{u=1}^{N}$ the set of predictions. Generally we have $M < N$, so we pad y to size N with $\emptyset$ (no object) and denote it by $\bar{y}$. The loss function, namely the Hungarian loss, is defined as:

$$\mathcal{L}_{\text{Hungarian}}(\bar{y}, \hat{y}) = \sum_{i=1}^{N} \left[\mathcal{L}_{class}^{i,\hat{\sigma}(i)} + \mathbb{1}_{\{\bar{y}_i \neq \emptyset\}} \mathcal{L}_{box}^{i,\hat{\sigma}(i)} \right] \quad (1)$$

where $\mathcal{L}_{class}^{i,\hat{\sigma}(i)}$ and $\mathcal{L}_{box}^{i,\hat{\sigma}(i)}$ are the classification loss and bounding box regression loss, respectively, between the ith ground truth and the $\hat{\sigma}(i)^{th}$ prediction. And $\hat{\sigma}$ is the optimal bipartite matching between padded ground-truth set $\bar{y}$ and prediction set $\hat{y}$ with lowest matching cost:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i=1}^{N} \mathcal{L}_{\text{match}}(\bar{y}_i, \hat{y}_{\sigma(i)}) \quad (2)$$

where $\mathfrak{S}_N$ is the set of all N permutations and $\mathcal{L}_{\text{match}}$ is a pair-wise matching cost.

DETR [3] uses an encoder-decoder Transformer [36] framework built upon the CNN backbone. The Transformer encoder part processes the flattened deep features[1] from the CNN backbone. Then, the non-autoregressive decoder part takes the encoder's outputs and a set of learned object query vectors as the input, and predicts the category labels and bounding boxes accordingly as the detection output. The cross-attention module plays an important role in the decoder by attending to different locations in the image for different object queries. We refer the reader unfamiliar with the Transformer concepts to the appendix. The attention mechanism in DETR eliminates the need for NMS post-processing because the self-attention component can learn to remove duplicated detection, *i.e.*, its Hungarian loss (Equation 1) encourages one target per object in the bipartite matching.

2.3. Improving Ground-truth Assignments

The Hungarian loss in DETR can be viewed as an end-to-end way to assign ground-truth labels to the system predictions. Prior to DETR, heuristic rules have been tried for this task [11, 30, 28]. There are a few other prior work that try to improve the heuristic ground-truth assignment rules. [43] formulates an MLE procedure to learn the matching between sliding windows and ground truth objects. [31] proposes a generalized IoU which provides a better metric. Nevertheless, those methods do not directly optimize a set-based objective and still require an NMS post-processing step.

2.4. Attention-based Object Detection

Attention-based modeling has been the current workhorse in the Natural Language Processing (NLP) domain [36, 10], and is becoming increasingly popular in recent object detection research. Before the invention of DETR, [15] proposes an attention-based module to model the relation between objects, which can be inserted into existing detectors and leads to better recognition and less duplication. [27] uses a Spatial Attention Module to re-weight feature maps for making foreground features standing out. [5] uses a Transformer-like attention-based module to bridge different forms of representations. But, none of those methods have tried an end-to-end set prediction objective.

3. What Causes the Slow Convergence of DETR?

To pin down the main factors we ran a set of experiments with DETR and its variants which are built on top of the

---

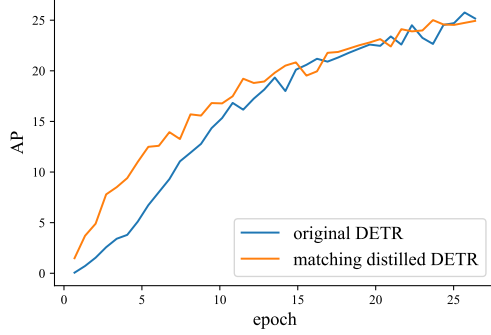[1] In this paper, we use "feature points" and "features" interchangeably.

Figure 1. AP results on COCO validation set: original DETR v.s. matching distilled DETR. We can see that matching distillation accelerates the training of DETR in the first few epochs.
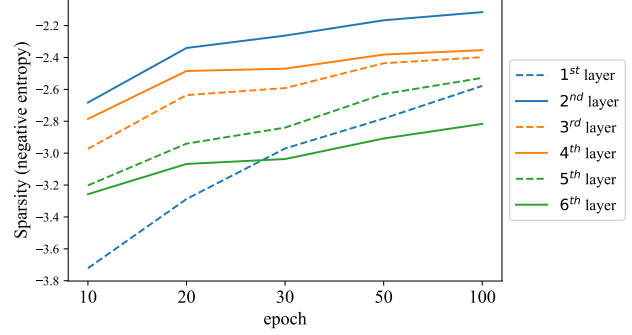

Figure 2. Sparsity (negative entropy) of Transformer cross-attention in each layer, obtained by evaluation on the COCO validation data. Different line styles represent different layers. We can see that the sparsity consistently increases, especially for the 1st cross-attention layer between encoder and decoder.

ResNet-50 backbone and evaluated on the COCO 2017 validation set.

3.1. Does Instability of the Bipartite Matching Affect Convergence?

As a unique component in DETR, the Hungarian loss based on the bipartite matching (Section 2.2) could be unstable due to the following reasons:

- The initialization of the bipartite matching is essentially random;
- The matching instability would be caused by noisy conditions in different training epochs.

To examine the effects of these factors, we propose a new training strategy for DETR, namely matching distillation. That is, we use a well pre-trained DETR as the teacher model, whose predicted bipartite matching is treated as the ground-truth label assignment for the student model. All stochastic modules in the teacher model (*i.e.*, dropout [33] and batch normalization [16]) are turned off to ensure the provided matching is deterministic, which eliminates the randomness and instability of the bipartite matching and hence in the Hungarian loss.

We evaluated both the original DETR and matching distilled DETR. Figure 1 shows the results with the first 25 epochs. We can see that the matching distillation strategy does help the convergence of DETR in the first few epochs. However, such effect becomes insignificant after around 15 epochs. This means that the instability in the bipartite matching component of DETR only contributes partially to the slow convergence (especially in the early training stage) but not necessarily the main reason.

3.2. Are the Attention Modules the Main Cause?

Another distinct part of DETR in comparison with other modern object detectors is its use of the Transformer modules, where the Transformer attention maps are nearly uniform in the initialization stage, but gradually become more and more sparse during the training process towards the convergence. Prior work [17] shows that replacing some attention heads in BERT [10] with sparser modules (*e.g.*, convolutions) can significantly accelerate its training. Therefore, it is natural for us to wonder how much the sparsity dynamics of Transformer attention modules in DETR contribute to its slow convergence.

In analyzing the effects of the DETR's attention modules on its optimization convergence, we focus on the sparsity dynamics of the cross-attention part in particular, because the cross-attention module is a crucial module where object queries in the decoder obtain object information from the encoder. Imprecise (under-optimized) cross-attention may not allow the decoder to extract accurate context information from images, and thus results in poor localization especially for small objects.

We collect the attention maps of cross-attention when evaluating the DETR model at different training stages. As attention maps can be interpreted as probability distributions, we use negative entropy as an intuitive measure of sparsity. Specifically, given a $n \times m$ attention map $\mathbf{a}$, we first calculate the sparsity of each source position $i \in [n]$ by $\frac{1}{m} \sum_{j=1}^{m} P(a_{i,j}) \log P(a_{i,j})$, where $a_{i,j}$ represents the attention score from source position i to target position j. Then we average the sparsities for all attention heads and all source positions in each layer. The masked positions [3] are not considered in the computation of sparsity.

Figure 2 shows the sparsities with respect to different epochs at several layers. we can see that the sparsity of cross-attention consistently increases and does not reach a plateau even after 100 training epochs. This means that the cross-attention part of DETR is more dominating a factor for the slow convergence, compared to the early-stage bipartite-matching instability factor we discussed before.
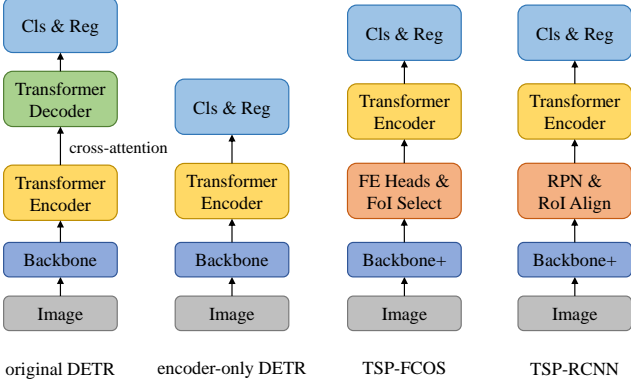
Figure 3. Illustration of original DETR, encoder-only DETR, TSP-FCOS, and TSP-RCNN, where Backbone+, FE Heads, RPN, Cls & Reg represents "Backbone + FPN", "Feature Extraction Heads (Subnets)", "Region Proposal Network", "Classification & Regression", respectively. A more detailed illustration of TSP-FCOS and TSP-RCNN can be found in Figure 5.
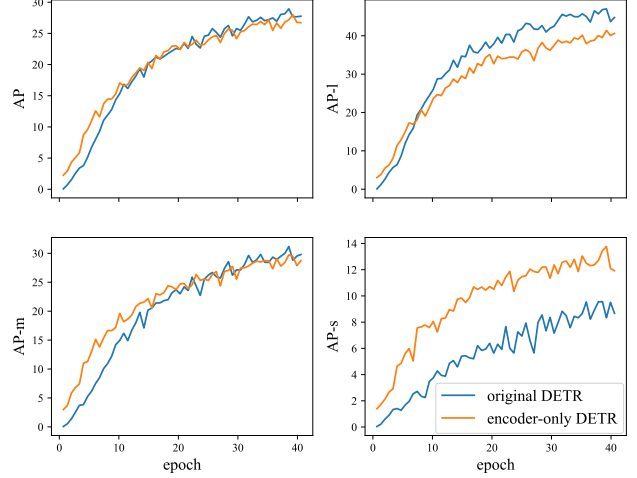

Figure 4. AP, AP-l, AP-m, and AP-s results on COCO validation set: original DETR v.s. encoder-only DETR. We can see that encoder-only DETR significantly accelerate the training of small object detection.

3.3. Does DETR Really Need Cross-attention?

Our next question is: Can we remove the cross-attention module from DETR for faster convergence but without sacrificing its prediction power in object detection? We answer this question by designing an encoder-only version of DETR and comparing its convergence curves with the original DETR.

In the original DETR, the decoder is responsible for producing the detection results (category label and bounding box) per object query. In contract, the encoder-only version of DETR (introduced by us) directly uses the outputs of Transformer encoder for object prediction. Specifically, for a $H \times W$ image with a $\frac{H}{32} \times \frac{W}{32}$ Transformer encoder feature map, each feature is fed into a detection head to predict a detection result. Since the encoder self-attention is essentially identical to the self-attention in a non-autoregressive decoder, a set prediction training is still feasible for encoder-only DETR. More details of encoder-only DETR can be found in the appendix. Figure 3 compares the original DETR and the encoder-only DETR, and two of our newly proposed models (TSP-FCOS and TSP-RCNN) which are described in the next section.

Figure 4 presents the the Average Precision (AP) curves of the original DETR and the encoder-only DETR, including the overall AP curve (denoted as AP) and the curves for large (AP-l), medium (AP-m), and small (AP-s) objects[2], respectively. The over-all curves (left upper corner) show that the encoder-only DETR performs as well as the original DETR. This means that we can remove the cross-attention part from DETR without much performance degeneration, which is a positive result. From the remaining curves we can see that the encoder-only DETR out-

performs the original DETR significantly on small objects and partly on medium object, but under-performs on large objects on the other hand. A potential interpretation, we think, is that a large object may include too many potentially matchable feature points, which are difficult for the sliding point scheme in the encoder-only DETR to handle. Another possible reason is that a single feature map processed by encoder is not robust for predicting objects of different scales [21].

4. The Proposed Methods

Based on our analysis in the previous section, for speeding up the convergence of DETR we need to address both the instability issue in the bipartite matching part of DETR and the cross-attention issue in Transformer modules. Specifically, in order to leverage the speed-up potential of encoder-only DETR we need to overcome its weakness in handling the various scales of objects. Recently, FCOS [34] (Fully Convolutional One-Stage object detector) shows that multi-level prediction with Feature Pyramid Network (FPN) [21] is a good solution to this problem. Inspired by this work we propose our first model, namely Transformer-based Set Prediction with FCOS (TSP-FCOS). Then based on TSP-FCOS, we further apply two-stage refinement, which leads to our second model, namely, Transformer-based Set Prediction with RCNN (TSP-RCNN).

4.1. TSP-FCOS

TSP-FCOS combines the strengths of both FCOS and encoder-only DETR, with a novel component namely Feature of Interest (FoI) selection which enables the Trans-

---

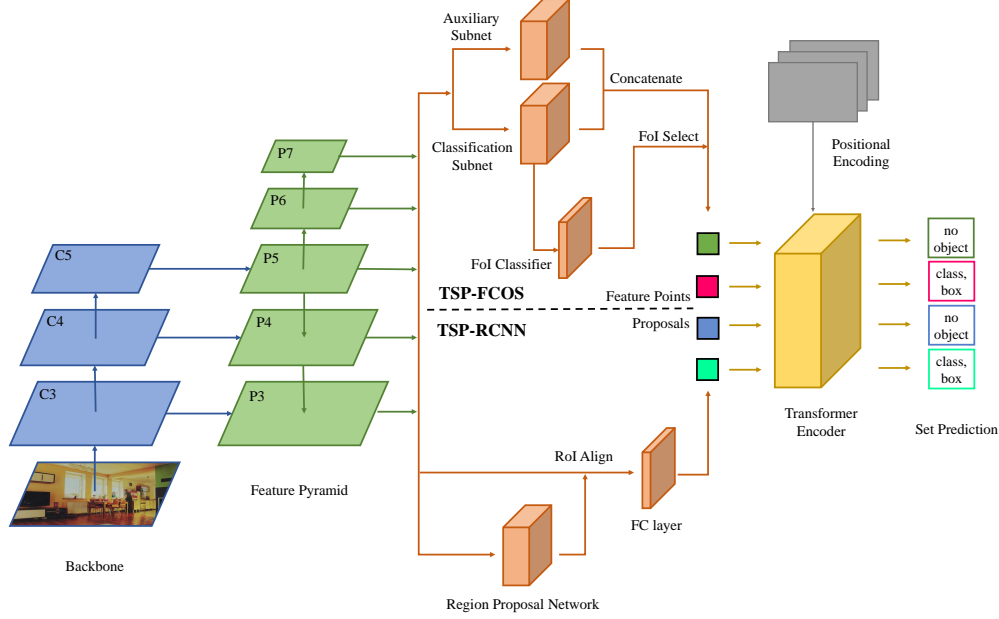[2]We follow the definitions of small, medium, and large objects in [23].

Figure 5. The network architectures of TSP-FCOS and TSP-RCNN, where C_3 to C_5 denote the feature maps of the backbone network and P_3 to P_7 of the Feature Pyramid Network (FPN). Both TSP-FCOS (upper) and TSP-RCNN (lower) are equipped with a Transformer encoder and trained with set prediction loss. The difference between them is that the FoI classifier in TSP-FCOS only predicts objectness of each feature (*i.e.*, Features of Interest), but the Region Proposal Network (RPN) in TSP-RCNN predicts both bounding boxes and their objectness as Regions of Interest (RoI), namely, proposals.

former encoder to handle multi-level features, and a new bipartite matching scheme for faster set prediction training. Figure 5 (upper part) illustrates the network architecture of TSP-FCOS, with the following components:

Backbone and FPN   We follow FCOS [34] on the design of the backbone and the Feature Pyramid Network (FPN) [21]. At the beginning of the pipeline, a backbone CNN is used to extract features from the input images. Based on the feature maps from the backbone, we build the FPN component which produces multi-level features that can help encoder-only DETR detect objects of various scales.

Feature extraction subnets   For a fair comparison with other one-stage detectors (*e.g.*, FCOS and RetinaNet), we follow their design and use two feature extraction heads shared across different feature pyramid levels. We call one of them classification subnet (head), which is used for FoI classification. The other is called auxiliary subnet (head). Their outputs are concatenated and then selected by FoI classifier.

Feature of Interest (FoI) classifier   In the self-attention module of Transformer, the computation complexity is quadratic to the sequence length, which prohibits directly using all the features on the feature pyramids. To improve the efficiency of self-attention, we design a binary classifier

to select a limited portion of features and refer them as Features of Interest (FoI). The binary FoI classifier is trained with FCOS's ground-truth assignment rule[3]. After FoI classification, top scored features are picked as FoIs and fed into the Transformer encoder.

Transformer encoder   After the FoI selection step, the input to Transformer encoder is a set of FoIs and their corresponding positional encoding. Inside each layer of Transformer encoder, self-attention is performed to aggregate the information of different FoIs. The outputs of the encoder pass through a shared feed forward network, which predicts the category label (including "no object") and bounding box for each FoI.

Positional encoding   Following DETR, we generalize the positional encoding of Transformer [36] to the 2D image scenario. Specifically, for a feature point with normalized position $(x, y) \in [0, 1]^2$, its positional encoding is defined as $[PE(x) : PE(y)]$, where $[:]$ denotes concatenation and function PE is defined by:

$$PE(x)_{2i} = \sin(x/10000^{2i/\mathrm{d_{model}}})$$
$$PE(x)_{2i+1} = \cos(x/10000^{2i/\mathrm{d_{model}}})$$

(3)

where $\mathrm{d_{model}}$ is the dimension of the FoIs.

---

[3]Please refer to the FCOS paper [34] for more details.

Faster set prediction training As mentioned in Section 2.2, the object detection task can be viewed as a set prediction problem. Given the set of detection results and ground truth objects, the set prediction loss links them together and provides an objective for the model to optimize. But as we show in Section 3.1, the Hungarian bipartite-matching loss can lead to slow convergence in the early stage of training. Therefore, we design a new bipartite matching scheme for faster set prediction training of TSP-FCOS. Specifically, a feature point can be assigned to a ground-truth object only when the point is in the bounding box of the object and in the proper feature pyramid level. This is inspired by the ground-truth assignment rule of FCOS [34]. Next, a restricted cost-based matching process (Equation 2) is performed to determine the optimal matching between the detection results and the ground truth objects in the Hungarian loss (Equation 1).

4.2. TSP-RCNN

Based on the design of TSP-FCOS and Faster RCNN, we can combine the best of them and perform a two-stage bounding box refinement as set prediction, which requires more computational resources but can detect objects more accurately. This idea leads to TSP-RCNN (Transformer-based Set Prediction with RCNN). Figure 5 (lower part) illustrates the network architecture of our proposed TSP-RCNN. The main differences between TSP-FCOS and TSP-RCNN are as follows:

Region proposal network In TSP-RCNN, instead of using two feature extraction heads and FoI classifier to obtain the input of Transformer encoder, we follow the design of Faster RCNN [30] and use a Region Proposal Network (RPN) to get a set of Regions of Interest (RoIs) to be further refined. Different from FoIs in TSP-FCOS, each RoI in TSP-RCNN contains not only an objectness score, but also a predicted bounding box. We apply RoIAlign [13] to extract the information of RoIs from multi-level feature maps. The extracted features are then flattened and fed into a fully connected network as the input of Transformer encoder.

Positional encoding The positional information of a RoI (proposal) is defined by four quantities (cx, cy, w, h), where $(cx, cy) \in [0, 1]^2$ denotes the normalized center coordinates and $(w, h) \in [0, 1]^2$ denotes the normalized height and width. We use $[PE(cx) : PE(cy) : PE(w) : PE(h)]$ as the positional encoding of the proposal, where PE and $[:]$ is defined in the same way as TSP-FCOS.

Faster set prediction training TSP-RCNN is also trained with a set prediction loss. Different from TSP-FCOS, we borrow the ground-truth assignment rule from Faster RCNN for faster set prediction training of TSP-RCNN. Specifically, a proposal can be assigned to a ground-truth object if and only if the intersection-over-union (IoU) score between their bounding boxes is greater than 0.5.

5. Experiments

5.1. Dataset and Evaluation Metrics

We evaluate our methods on the COCO [23] object detection dataset, which includes 80 object classes. Following the common practice [22, 34], we use all $115k$ images in `trainval35k` split for training and all $5k$ images in `minival` split for validation. The test result is obtained by submitting the results of `test-dev` split to the evaluation server. For comparison with other methods, we mainly focus on the Average Precision (AP), which is the primary challenge metric used in COCO, and FLOPs, which measures the computation overhead.

5.2. Implementation Details

We briefly describe the default settings of our implementation. More detailed settings can be found in appendix.

Backbone We use ResNet-50 and ResNet-101 [14] as the backbone, and a Feature Pyramid Network [21] 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 [45] in the last three stages of ResNet. The feature maps have 256 channels.

TSP-FCOS Following FCOS [34], both classification subnet and auxiliary subnet use four 3×3 convolutional layers with 256 channels and group normalization [37]. In FoI selection, we select top 700 scored feature positions from FoI classifier as the input of Transformer encoder.

TSP-RCNN Different from the original Faster RCNN, we apply 2 unshared convolutional subnets to P_3-P_7 as classification and regression heads of RPN and use a RetinaNet [22] style anchor generation scheme. We find this improves the performance of RPN with less computation overhead. In RoI selection, we select top 700 scored features from RPN. RoI Align operation [13] and a fully connected layer are applied to extract the proposal features from RoIs.

Transformer encoder As both TSP-FCOS and TSP-RNN only have a Transformer encoder while DETR has both Transformer encoder and decoder, to be comparable in terms of FLOPs with DETR-DC5, we use a 6-layer Transformer encoder of width 512 with 8 attention heads. The hidden size of feed-forward network (FFN) in Transformer

Model	Backbone	Epochs	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$	FLOPs	FPS
FCOS†	ResNet-50	36	41.0	59.8	44.1	26.2	44.6	52.2	177G	17
Faster RCNN-FPN	ResNet-50	36	40.2	61.0	43.8	24.2	43.5	52.0	180G	19
Faster RCNN-FPN+	ResNet-50	108	42.0	62.1	45.5	26.6	45.4	53.4	180G	19
DETR+	ResNet-50	500	42.0	62.4	44.2	20.5	45.8	**61.1**	86G	21
DETR-DC5+	ResNet-50	500	43.3	63.1	45.9	22.5	47.3	**61.1**	187G	7
TSP-FCOS	ResNet-50	36	43.1	62.3	47.0	26.6	46.8	55.9	189G	15
TSP-RCNN	ResNet-50	36	43.8	63.3	48.3	28.6	46.9	55.7	188G	11
TSP-RCNN+	ResNet-50	96	**45.0**	**64.5**	**49.6**	**29.7**	**47.7**	58.0	188G	11
FCOS†	ResNet-101	36	42.5	61.3	45.9	26.0	46.5	53.6	243G	13
Faster RCNN-FPN	ResNet-101	36	42.0	62.5	45.9	25.2	45.6	54.6	246G	15
Faster RCNN-FPN+	ResNet-101	108	44.0	63.9	47.8	27.2	48.1	56.0	246G	15
DETR+	ResNet-101	500	43.5	63.8	46.4	21.9	48.0	61.8	152G	15
DETR-DC5+	ResNet-101	500	44.9	64.7	47.7	23.7	49.5	**62.3**	253G	6
TSP-FCOS	ResNet-101	36	44.4	63.8	48.2	27.7	48.6	57.3	255G	12
TSP-RCNN	ResNet-101	36	44.8	63.8	49.2	29.0	47.9	57.1	254G	9
TSP-RCNN+	ResNet-101	96	**46.5**	**66.0**	**51.2**	**29.9**	**49.7**	59.2	254G	9

Table 1. Evaluation results on COCO 2017 validation set. † represents our reproduction results. + represents that the models are trained with random crop augmentation and a longer training schedule. We use Detectron2 package to measure FLOPs and FPS. A single Nvidia GeForce RTX 2080 Ti GPU is used for measuring inference latency.

Model	AP	AP$_S$	AP$_M$	AP$_L$
TSP-RCNN-R50	**43.8**	**28.6**	**46.9**	55.7
w/o set prediction loss	42.7	27.6	45.5	**56.2**
w/o positional encoding	43.4	28.4	46.3	55.0
TSP-RCNN-R101	**44.8**	**29.0**	**47.9**	57.1
w/o set prediction loss	44.0	27.6	47.2	**57.1**
w/o positional encoding	44.4	28.2	47.7	56.7

Table 2. Evaluation results on COCO 2017 validation set for ablation study of set prediction loss and positional encoding.

Model	AP	AP$_S$	AP$_M$	AP$_L$
FCOS	45.3	28.1	49.0	59.3
TSP-FCOS	**46.1**	**28.5**	**49.7**	**60.2**
Faster-RCNN	44.1	26.4	47.6	58.1
TSP-RCNN	**45.8**	**29.4**	**49.2**	**58.4**

Table 3. Evaluation results on COCO 2017 validation set with ResNet-101-DCN backbone.

is set to 2048. During training, we randomly drop 70% inputs of Transformer encoder to improve the robustness of set prediction.

Training We follow the default setting of Detectron2 [38], where a 36-epoch (3×) schedule with multi-scale train-time augmentation is used.

5.3. Main Results

Table 1 shows our main results on COCO 2017 validation set. We compare TSP-FCOS and TSP-RCNN with FCOS [34], Faster RCNN [30], and DETR [3]. From the table, we can see that our TSP-FCOS and TSP-RCNN sig-

nificantly outperform original FCOS and Faster RCNN. Besides, we can find that TSP-RCNN is better than TSP-FCOS in terms of overall performance and small object detection but slightly worse in terms of inference latency.

To compare with state-of-the-art DETR models, we use a similar training strategy in DETR [3], where a 96-epoch (8×) training schedule and random crop augmentation is applied. We denote the enhanced version of TSP-RCNN by TSP-RCNN+. We also copy the results of enhanced Faster RCNN (*i.e.*, Faster RCNN+) from [3]. Comparing these models, we can find that our TSP-RCNN obtains state-of-the-art results with a shorter training schedule. We also find that TSP-RCNN+ still under-performs DETR-DC5+ on large object detection. We think this is because of the inductive bias of the encoder-decoder scheme used by DETR and its longer training schedule.

5.4. Model Analysis

For model analysis, we evaluate several models trained in our default setting, *i.e.*, with a 36-epoch (3×) schedule and without random crop augmentation.

5.4.1 Ablation study

We conduct an ablation study of set prediction loss and positional encoding, which are two essential components in our model. Table 2 show the results of ablation study for TSP-RCNN with ResNet-50 and ResNet-101 backbone. From the table, we can see that both set prediction loss and positional encoding are very important to the success of our TSP mechanism, while set prediction loss contributes more than positional encoding to the improvement of TSP-RCNN.

Model	Backbone	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
RetinaNet [22]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
FSAF [44]	ResNet-101	40.9	61.5	44.0	24.0	44.2	51.3
FCOS [34]	ResNet-101	41.5	60.7	45.0	24.4	44.8	51.6
MAL [18]	ResNet-101	43.6	62.8	47.1	25.0	46.9	55.8
RepPoints [39]	ResNet-101-DCN	45.0	66.1	49.0	26.6	48.6	57.5
ATSS [42]	ResNet-101	43.6	62.1	47.4	26.1	47.0	53.6
ATSS [42]	ResNet-101-DCN	46.3	64.7	50.4	27.7	49.8	58.4
Faster RCNN [30]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2
Fitness NMS [35]	ResNet-101	41.8	60.9	44.9	21.5	45.0	57.5
Libra RCNN [26]	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 [20]	ResNet-101-DCN	46.8	**67.6**	51.5	28.0	**51.2**	**60.5**
TSD [32]	ResNet-101	43.2	64.0	46.9	24.0	46.3	55.8
Dynamic RCNN [41]	ResNet-101	44.7	63.6	49.1	26.0	47.4	57.2
Dynamic RCNN [41]	ResNet-101-DCN	46.9	65.9	51.3	28.1	49.6	60.0
TSP-RCNN	ResNet-101	<u>46.6</u>	<u>66.2</u>	<u>51.3</u>	<u>28.4</u>	<u>49.0</u>	<u>58.5</u>
TSP-RCNN	ResNet-101-DCN	**47.4**	66.7	**51.9**	**29.0**	49.7	59.1

Table 4. Comparison with state-of-the-art models on COCO 2017 test set (single-model and single-scale results). <u>Underlined</u> and **bold** numbers represent the best model with ResNet-101 and ResNet-101-DCN backbone, respectively.
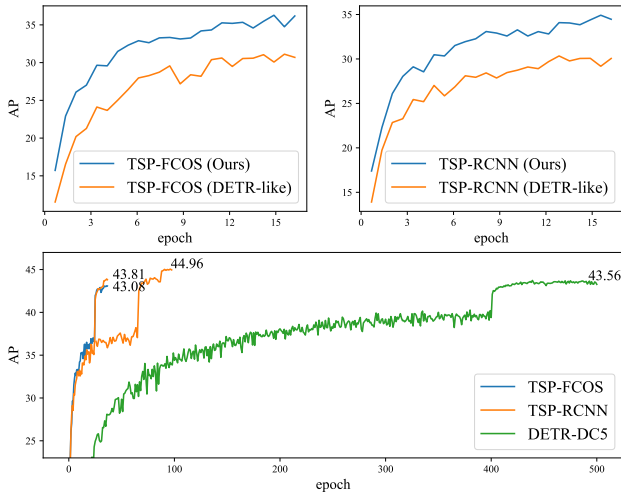

Figure 6. Top two plots compare the convergence speed of our proposed faster set prediction training loss and DETR-like loss for TSP-FCOS and TSP-RCNN. The bottom plot shows the convergence curves of TSP-FCOS, TSP-RCNN, and DETR-DC5, where for TSP-RCNN, we continue its training for additional 72 epochs with random crop augmentation from the 24th epoch. AP score leaps are caused by learning rate reduction.

5.4.2 Compatibility with deformable convolutions

One may wonder whether Transformer encoder and deformable convolutions [9, 45] are compatible with each other, as both of them can utilize long-range relation between objects. In Table 3, we compare TSP-FCOS and TSP-RCNN to FCOS and Faster RCNN with deformable ResNet-101 as backbone. From the results, we can see that the TSP mechanism is well complementary with de-

formable convolutions.

5.4.3 Analysis of convergence

We compare the convergence speed of our proposed faster set prediction training and DETR's original set prediction training in the upper part of Figure 6. We can see that our proposed faster training technique consistently accelerates the convergence of both TSP-FCOS and TSP-RCNN.

We also plot the convergence curves of TSP-FCOS, TSP-RCNN, and DETR-DC5 in the lower part of Figure 6, from which we can find that our proposed models not only converge faster, but also achieve better detection performance.

5.5. Comparison with State-of-the-Arts

We compare our proposed TSP-RCNN with multiple one-stage and two-stage object detection models [30, 35, 2, 32, 22, 44, 34, 4, 19, 39, 42] that also use ResNet-101 backbone or its deformable convolution network (DCN) [45] variant in Table 4. A 8× 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. Among all detectors with ResNet-101 or ResNet-101-DCN backbone, our model achieves the highest AP scores.

6. Conclusion

Aiming to accelerate the training convergence of DETR as well as to improve prediction power in object detection, we present an investigation on the causes of its slow convergence through extensive experiments, and propose two novel solutions, namely TSP-FCOS and TSP-RCNN, which

require much less training time and achieve the state-of-the-art detection performance. For future work, we would like to investigate the successful use of sparse attention mechanism [6, 8, 40] for directly modeling the relationship among multi-level features.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 11

[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. 8, 13

[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, 7, 11, 12

[4] Yuntao Chen, Chenxia Han, Naiyan Wang, and Zhaoxiang Zhang. Revisiting feature alignment for one-stage object detection. *arXiv preprint arXiv:1908.01570*, 2019. 8, 13

[5] Cheng Chi, Fangyun Wei, and Han Hu. Relationnet++: Bridging visual representations for object detection via transformer decoder. *Advances in Neural Information Processing Systems*, 33, 2020. 2

[6] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 9

[7] 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. 11

[8] Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019. 9

[9] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 8

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 3

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2, 11

[12] 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. 11

[13] 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. 1, 2, 6, 12

[14] 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. 6, 11

[15] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018. 2

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[17] Zi-Hang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Convbert: Improving bert with span-based dynamic convolution. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[18] 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. 8, 13

[19] 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. 8, 13

[20] 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. 8, 13

[21] 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. 1, 4, 5, 6, 12

[22] 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. 1, 2, 6, 8, 12, 13

[23] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 1, 4, 6

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 2

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 12

[26] 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. 8, 13

[27] Zheng Qin, Zeming Li, Zhaoning Zhang, Yiping Bao, Gang Yu, Yuxing Peng, and Jian Sun. Thundernet: Towards real-time generic object detection on mobile devices. In *Proceed-

ings of the IEEE International Conference on Computer Vision, pages 6718–6727, 2019. 2

[28] 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. 1, 2

[29] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 1

[30] 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, 2, 6, 7, 8, 11, 13

[31] 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, 12

[32] 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. 8, 13

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 3

[34] 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. 1, 4, 5, 6, 7, 8, 12, 13

[35] 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. 8, 13

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1, 2, 5

[37] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 6

[38] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019. 7, 12

[39] 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. 8, 13

[40] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 2020. 9

[41] 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. 8, 13

[42] 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. 1, 8, 13

[43] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. Freeanchor: Learning to match anchors for visual object detection. In *Advances in Neural Information Processing Systems*, pages 147–155, 2019. 2

[44] 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. 8, 13

[45] 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. 6, 8, 13

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 [7]. 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:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{\text{model}}}}\right) \cdot V, \quad (4)$$

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.

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:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2, \quad (5)$$

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.
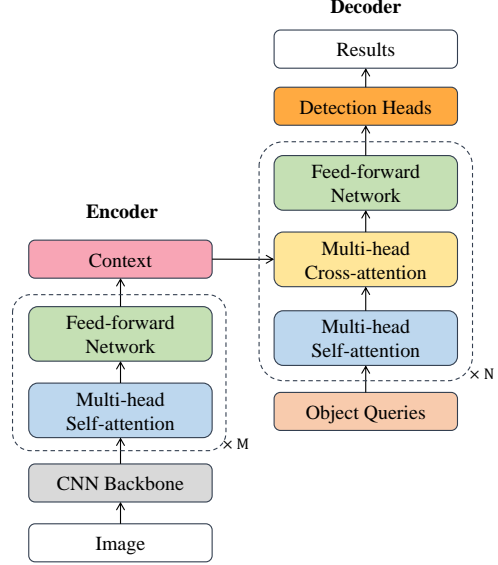

Figure 7. A detailed illustration of the DETR architecture. Residual connection and layer normalization are omitted.

Stacking Multi-head attention and feed-forward network are stacked alternately to form the encoder and the decoder, with residual connections [14] and layer normalization [1]. Figure 7 shows a detailed illustration of the DETR architecture.

A.2. Faster R-CNN

Faster R-CNN [30] is a two-stage object detection model, developed based on previous work of R-CNN [12] and Fast R-CNN [11]. 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 [13] 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) [34] 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 [21], 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.

Data augmentation   We follow the default setting of Detectron2 [38] 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 [31] losses for regression. Focal loss [22] 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 [25] 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 for the Transformer part, with a maximal L_2 gradient norm of 0.1.

Longer training schedule   We also use a 96-epoch ($8\times$) schedule in the paper. The 96-epoch ($8\times$) schedule will resume from 36-epoch ($3\times$) schedule's model checkpoint in the 24th 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 $8\times$ 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	Backbone	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
Faster RCNN [30]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2
Fitness NMS [35]	ResNet-101	41.8	60.9	44.9	21.5	45.0	57.5
Libra RCNN [26]	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 [20]	ResNet-101-DCN	46.8	67.6	51.5	28.0	51.2	60.5
TSD [32]	ResNet-101	43.2	64.0	46.9	24.0	46.3	55.8
Dynamic RCNN [41]	ResNet-101	44.7	63.6	49.1	26.0	47.4	57.2
Dynamic RCNN [41]	ResNet-101-DCN	46.9	65.9	51.3	28.1	49.6	60.0
RetinaNet [22]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
FSAF [44]	ResNet-101	40.9	61.5	44.0	24.0	44.2	51.3
FCOS [34]	ResNet-101	41.5	60.7	45.0	24.4	44.8	51.6
MAL [18]	ResNet-101	43.6	62.8	47.1	25.0	46.9	55.8
RepPoints [39]	ResNet-101-DCN	45.0	66.1	49.0	26.6	48.6	57.5
ATSS [42]	ResNet-101	43.6	62.1	47.4	26.1	47.0	53.6
ATSS [42]	ResNet-101-DCN	46.3	64.7	50.4	**27.7**	**49.8**	58.4
TSP-FCOS	ResNet-101	<u>46.1</u>	<u>65.8</u>	<u>50.3</u>	<u>27.3</u>	<u>49.0</u>	<u>58.2</u>
TSP-FCOS	ResNet-101-DCN	**46.8**	**66.4**	**51.0**	27.6	49.5	**59.0**

Table 5. Compare TSP-FCOS with state-of-the-art models on COCO 2017 test set (single-model and single-scale results). <u>Underlined</u> 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 6. Evaluation results on COCO 2017 validation set of models under various FLOPs. ResNet-50 is used as backbone.

D. 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 6 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 6, 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.

E. 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 [30, 35, 2, 32, 22, 44, 34, 4, 19, 39, 42] that also use ResNet-101 backbone or its deformable convolution network (DCN) [45] variant in Table 5. 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 compare Table 4 and Table 5, we can also find that TSP-FCOS slightly under-performs our proposed TSP-RCNN model.