DESTR: Object Detection with Split Transformer

Liqiang He and Sinisa Todorovic
Oregon State University
Corvallis, OR 97330, USA
{heli,sinis}@oregonstate.edu

Abstract

Self- and cross-attention in Transformers provide for high model capacity, making them viable models for object detection. However, Transformers still lag in performance behind CNN-based detectors. This is, we believe, because: (a) Cross-attention is used for both classification and bounding-box regression tasks; (b) Transformer's decoder poorly initializes content queries; and (c) Self-attention poorly accounts for certain prior knowledge which could help improve inductive bias. These limitations are addressed with the corresponding three contributions. First, we propose a new Detection Split Transformer (DESTR) that separates estimation of cross-attention into two independent branches – one tailored for classification and the other for box regression. Second, we use a mini-detector to initialize the content queries in the decoder with classification and regression embeddings of the respective heads in the mini-detector. Third, we augment self-attention in the decoder to additionally account for pairs of adjacent object queries. Our experiments on the MS-COCO dataset show that DESTR outperforms DETR and its successors.

1. Introduction

This paper addresses a basic vision problem, that of object detection in images. Recently proposed DEtection TRansformer (DETR) [2] and its successors, such as, e.g., Conditional DETR (C-DETR) [19] and Anchor DETR [25], have been demonstrated as competitive performers on the benchmark MS-COCO dataset [15], despite using a simpler backbone network with single-scale features than more complex state-of-the-art (SOTA) CNNs [20, 22, 24, 27]. This success has been ascribed to Transformers' high model capacity, since they estimate self-attention and cross-attention, thereby explicitly capturing relationships between parts and larger spatial contexts in the image. However, recent findings [3, 5] suggest that Transformers lack certain inductive biases possessed by CNNs to help them constrain the hypothesis space. Therefore, Transformers require longer training (e.g., DETR needs 500 training epochs) and larger amount of training data to compensate. In this work, we identify and address three key limitations of the mentioned family of detector Transformers, and thus improve their inductive bias.

The first limitation concerns cross-attention. DETR's decoder computes cross-attention between the encoder's output embedding and a set of learnable object queries, for estimating relationships between these queries and the entire image context. This cross-attention is then used for both classification and bounding-box regression of the queries. The same holds for other Transformer detectors.

Motivated by the success of FCOS detector [24] that splits the classification and box regression heads, as our first contribution, we propose to split estimation of cross-attention into two independent branches – one for classification, and the other for regression. Hence the name of our new Detection Split Transformer (DESTR). Since the branches will not share weights, each will likely focus on a different set of optimal features, as desired, rather than jointly use suboptimal features for both classification and regression. This is illustrated in Fig. 1, where we show cross-attention maps computed by DETR, C-DETR, and our DESTR. As can be seen, DETR's cross-attention appears to focus on most discriminative object parts which may not necessarily be informative for box regression. On the other hand, C-DETR's cross-attention seems to primarily focus on shape cues. DESTR's classification and regression cross-attention maps differ, as intended, where the former highlights more class-characteristic regions to help classification, and the latter has higher values on horizontal and vertical edges in the image to guide the box prediction, and looks more dilated to the same regions.

The second limitation concerns the poor initialization and need for long training of content queries in the decoder. Features of the learnable object queries in the decoder consist of the content embedding (a.k.a. content query) and positional embedding (a.k.a. spatial query). DETR learns the positional embedding so it captures a spatial distribution of objects in training images [2]. However, the content queries

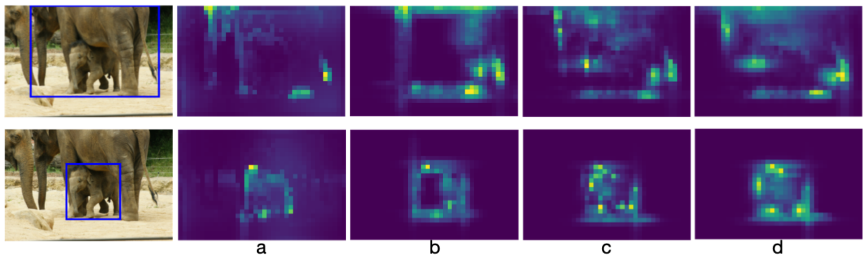

Figure 1. Cross-attention maps estimated by: (a) DETR trained with 500 training epochs, and (b) C-DETR trained with 50 epochs. (c) DESTR's classification cross-attention focuses on discriminative object parts. (d) DESTR's regression cross-attention is more dilated to the same region comparing to (c). DESTR is trained with 50 epochs. R50 is used as backbone for all these three models. For better visualization, the figure shows square-root values of cross-attention. Warmer colors indicate higher cross-attention values.

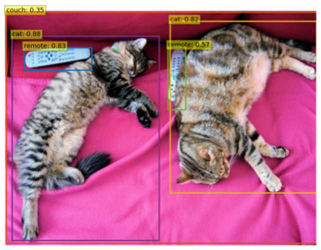

Figure 2. Pair self-attention: Since the remotes occur next to the cats and couch, we increase their attention by considering attention of pairs $\big($(cat1, remote1),(remote2, cat2)$\big)$, and $\big($(couch, remote1),(couch, remote2)$\big)$.

are inferred for every image from scratch. This makes training difficult due to the higher dependence of cross-attention on the content than spatial query [2, 19], especially in the initial stages of training when the content queries are not "strong" enough to match well the positional embedding of the keys. To address these issues C-DETR [19]: (a) separates the content and positional dot-products when computing cross-attention, thereby relaxing their inter-dependence; and (b) conditions the positional embedding of each query with the corresponding decoder output embedding of the previous stage, and thus constrains the content to focus more on discriminative regions within the previously predicted bounding box of the query. We adopt these modifications of C-DETR, since they enable a significant decrease of training epochs, and further extend the framework with the remaining two contributions.

As our second contribution, we propose to learn not only the positional embedding, but also the content embedding, and to do so by following our first contribution – i.e., to learn the content as the separate classification and regression embeddings. To this end, we insert a mini-detector after the encoder to predict a set of initial object proposals. Features of these object candidates, used by the mini-detector's classification and regression heads, can be passed to the decoder and thus initialize the classification and regression queries, instead of inferring them from scratch. Our grounding of object queries on initial object proposals is expected to facilitate training. Importantly, this will also enable DESTR to consider a flexible number of object queries in both training and testing, rather than use a predefined fixed number as in DETR and C-DETR.

Finally, as out third contribution, we seek to incorporate certain prior knowledge in self-attention of the decoder, and in this way better constrain the hypothesis space. We expect that object instances occur within similar surrounding spatial contexts, which could improve estimation of self-attention in the decoder. Therefore, instead of computing the common self-attention for every query in isolation, we compute *pair self-attention* for every two pairs of queries, where each pair has been predicted to be spatially adjacent by the previous decoder stage. That is, we condition our pair self-attention on the corresponding decoder output of the previous stage. The example in Fig. 2 illustrates advantages of the proposed strategy. One of the remote controllers is partially occluded, and, therefore, provides a low attention to another fully visible remote. But since both remotes occur next to the cats and couch, we could increase their attention by considering attention of pairs $\big($(left cat, left remote), (right remote, right cat)$\big)$ and $\big($(couch, left remote), (couch, right remote)$\big)$. We do so without increasing complexity to quadratic in the number of queries.

In summary, our three main contributions include:

1. The classification and regression branches in the decoder compute separately their respective cross-attentions, instead of sharing the same cross-attention;
2. A mini-detector is inserted after the encoder for learning classification, regression and positional embeddings. Embeddings of the object queries in the decoder are initialized with the corresponding classification, regression and positional embeddings of object proposals predicted by the mini-detector;
3. Pair self-attention of the queries and their adjacent spatial contexts is estimated in the decoder, instead of the common self-attention for every individual query.

Our experiments demonstrate that DESTR outperforms C-DETR and other recent Transformer detectors with a significant margin on MS-COCO-val [15]. DESTR is also a competitive performer relative to recent CNN-based detectors; although, a direct comparison with CNNs is unfair, since they typically use multi-scale features and more complex backbone networks.

In the following, Sec. 2 reviews related work, Sec. 3 specifies DESTR, Sec. 4 presents our experimental evaluation, and Sec 5 concludes the paper.

2. Related work

Object detection is a long-standing problem, and reviewing the relevant literature is beyond our scope. We focus on discussing the most related work.

Anchor-free approaches [7, 10–13, 21, 21, 24, 26, 28, 29] have gained much interest, because of their relatively simple architecture and superiority in performance. They replace hand-crafted anchor boxes of anchor-based methods with reference points. For example, CornerNet [11] first predicts most likely corner-points, and then groups them with Associative Embedding. Alternatively, CenterNet [7] regresses the object centers. These two strategies are advanced in FCOS [24] that directly regresses bounding boxes to reference points and predicts their centerness on the multi-scale feature maps from FPN [14]. In addition, FCOS uses two separate FCN branches [17] for classification and regression, thereby explicitly learning separate features for each of the two tasks.

Inspired by FCOS, we disentangle the content embedding of DETR and C-DETR into the classification embedding and separate regression embedding, by the means of splitting estimation of cross-attention into the classification and regression branches which do not share parameters. In addition, we use one-scale FCOS as the mini-detector.

Transformer-based detectors [2, 8, 19, 23, 25, 30] cast object detection as a direct set prediction problem. In comparison with the aforementioned anchor-free CNNs, Transformer detectors have a more streamlined architecture and do not require heuristic post-processing, such as, e.g., non-maximum suppression (NMS). However, DETR [2] requires very long training time. A number of DETR variants [8, 19, 23, 25, 30] have addressed this issue. For example, Deformable DETR [30] replaces the global dense attention with deformable one, so that it only needs to attend to a small set of sampling points from multi-scale image features. However, Deformable DETR requires additional learning of offsets for point sampling. TSP-FCOS or TSP-RCNN [23] removes cross-attention from the decoder, and uses instead the detection head of FCOS or R-CNN. This, however, results in a hybrid architecture that is not well streamlined. Anchor DETR [25] encodes anchor points as the object queries, but they require a predefined set of anchors. SMCA [8] initially predicts the centers and scales of candidate objects for generating a Gaussian-weighted spatial map of object locations, which is then used to constrain estimation of cross-attention to focus more on high values in the spatial map. We find their constraint too strong, because the initial object proposals may not be accurately detected, and also cross-attention usually requires larger spatial extents for reasoning (see Fig. 1).

C-DETR [19] takes another direction toward reducing training time. For each query, it learns a conditional positional embedding from the corresponding previous decoder output embedding. This motivates us to additionally condition the content embedding on features of object proposals which have predicted by the mini-detector, instead of inferring the content embedding from scratch as in C-DETR. Thus, unlike C-DETR, we enable learning of both content and positional embeddings, and use the decoder as a context-guided refinement module.

The literature also presents other Transformer-based detectors, such as ViT [6] and SWIN [16] which make contributions in the encoder part, aimed at effectively enhancing the feature embedding with the entire image content. In contrast, our three contributions are related to the decoder.

3. Specification of DESTR

Fig. 3 shows an overview of our DESTR. The encoder remains the same as in DETR. A mini-detector, equipped with a classification branch and box regression branch, is inserted between the encoder and decoder for initial prediction of object candidates. The output classification, regression, and positional embeddings of the mini-detector are passed to the detector to initialize the corresponding three types of embeddings of the object queries. Note that the object proposals do not serve as anchors, and they are not used for any specific constraints in the decoder. Their features are only used for initialization of the object queries,

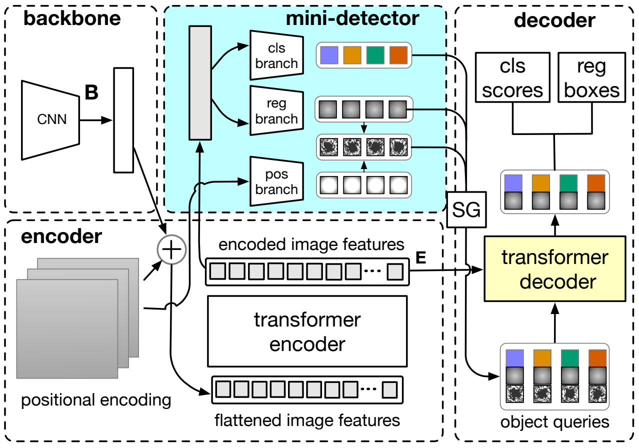

Figure 3. An overview of DESTR. DETR is extended with a mini-detector (the cyan block) whose learned embeddings of the classification (cls) and box regression (reg) branches are passed to the decoder for initializing the object queries. The decoder splits cross-attention for classification (cls) and regression (reg). As in C-DETR, the positional embedding (pos) is conditioned on the decoder's previous output. The stop gradient (SG) is applied before the mini-detector's output is passed to the decoder. All components of DESTR are trained end-to-end.

which are then further refined in the decoder. The decoder splits computation of cross-attention into the classification and regression branches. As in C-DETR, the positional embedding is conditioned on the output embedding of the decoder's regression branch from the previous stage.

In the following, we specify the mini-detector and certain modules in the decoder that represent our contributions. Since the remaining components of DESTR are the same as in DETR and C-DETR, their description is omitted.

3.1. The Mini-detector

The mini-detector is aimed at initial object detection which will be further refined by the decoder. Therefore, to keep model complexity in check, the mini-detector has a relatively simple architecture – much simpler than existing detectors (e.g., FCOS).

As input, the mini-detector takes enhanced features of the encoder's last layer, $F \in \mathbb{R}^{H\times W\times C}$, where H, W denote the feature map size, and C is the number of channels. As in FCOS [24], the mini-detector predicts an object for every cell feature $\mathbf{f}_{(i,j)}$ of F at location (i, j). The object prediction consists of classification $\mathbf{c}_{(i,j)}$ and regression of the bounding box center (b_{cx}, b_{cy}), height b_h, and width b_w, $\mathbf{b}_{(i,j)} = [b_{cx}, b_{cy}, b_w, b_h]$, defined as

$$\begin{aligned}\mathbf{c}_{(i,j)} &= \text{sigmoid}(\text{FFN}^{\text{cls}}(\text{FCN}^{\text{cls}}(\mathbf{f}_{(i,j)}))),\\ \mathbf{b}_{(i,j)} &= \text{sigmoid}(\text{FFN}^{\text{reg}}(\text{FCN}^{\text{reg}}(\mathbf{f}_{(i,j)})) + [\mathbf{s}_{(i,j)}^{\top}, 0, 0]),\\ \mathbf{s}_{(i,j)} &= \text{FCN}^{\text{pos}}(\mathbf{p}_{(i,j)}),\ \mathbf{p}_{(i,j)} = \text{sinusoidal}([i\ , j]),\end{aligned} \tag{1}$$

where FCN^{cls} and FCN^{reg} denote two separate four-layer fully convolutional networks followed by a classification head and box head, respectively; $\mathbf{s}$ denotes unnormalized 2D coordinates of the reference point for each cell, estimated by another four-layer FCN which embeds the positional encoding of each cell $\mathbf{p} \in P$, $P \in \mathbb{R}^{H\times W\times C}$.

As in [2], the predicted objects are supervised with a set-based loss that forces the one-to-one correspondence between the predictions $\hat{y} = \{\hat{y}_n = (\mathbf{c}_n, \mathbf{b}_n) : n = 1, \ldots, N\}$ and the ground-truth set of objects y via bipartite matching. After the bipartite matching, each matched prediction is supervised with the standard loss defined as a linear combination of a negative log-likelihood for class prediction and a box loss, as in [2]. As shown in Fig. 3, to avoid over-fitting, we apply the stop-gradient operation before the mini-detector's output is passed to the decoder.

Since the mini-detector makes predictions at every cell location (i, j) of F, we select a subset of K predictions with the highest classification scores as the initial object proposals. Embeddings of these K proposals are passed to the decoder to initialize the corresponding K object queries as

$$\begin{aligned}\mathbf{e}_n &= \text{cat}(\text{FCN}^{\text{cls}}(\mathbf{f}_n), \text{FCN}^{\text{reg}}(\mathbf{f}_n)) \in \mathbb{R}^{2C},\\ \mathbf{p}_n &= \text{sinusoidal}([\hat{b}_{n,cx}, \hat{b}_{n,cy}]) \in \mathbb{R}^{2C},\end{aligned} \tag{2}$$

where the content embedding $\mathbf{e}_n$ concatenates the classification and regression embeddings of the nth object proposal, and the positional embedding $\mathbf{p}_n$ projects the predicted box center of the nth object proposal onto the 256-dimensional sinusoidal embedding space.

3.2. Our Decoder

Fig. 4 shows differences between our decoder and C-DETR's decoder. As can be seen, the first difference is that we divide the content embedding in C-DETR into the classification and regression embeddings of the object queries. Second, we initialize these embeddings with the corresponding output of the mini-detector, whereas the content embedding in C-DETR is initially set to zero. Third, we split cross-attention for classification and regression, so each cross-attention could focus better on features relevant for the respective task. Finally, fourth, instead of estimating the common self-attention for each individual query, we estimate pair self-attention.

In the following, we specify our pair self-attention and split cross-attention.

3.2.1 Pair Self-attention

As in DETR, for every object query a, we first compute query $\mathbf{q}_a$, key $\mathbf{k}_a$, and value $\mathbf{v}_a$. The query and key are defined as a sum of the linearly projected content embedding $\mathbf{e}_a$ and positional embedding $\mathbf{p}_a$, whereas the value is

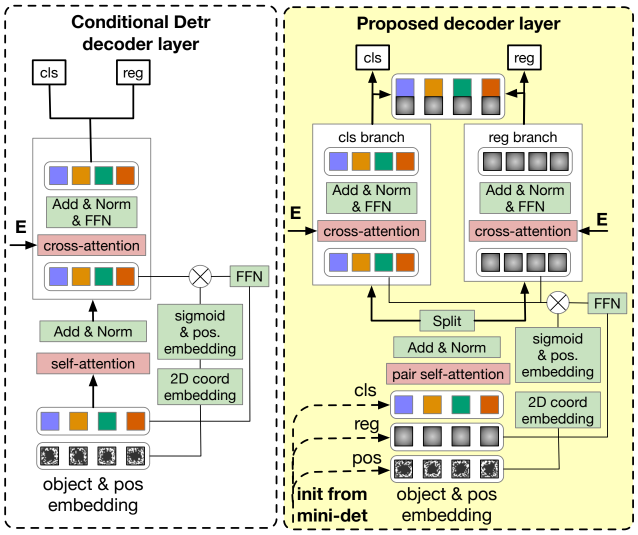

Figure 4. The four differences between C-DETR's decoder (left) and our decoder (right): (1) we divide the content embedding into the classification (cls) and regression (reg) embeddings; (2) the queries are initialized with the corresponding output of the mini-detector (mini-det); (3) cross-attention is split into two branches; (4) we extend self-attention to pair self-attention.

a linear projection of $\mathbf{e}_a$. Then, we estimate self-attention between every two object queries a and b as

$$A_1(a,b) = \mathbf{q}_a^\top \mathbf{k}_b, \tag{3}$$

and the self-attention output embedding of every query a as

$$\mathbf{o}_1(a) = \sum_{b \in \text{queries}} \text{softmax}\left(\frac{A_1(a,b)}{\sqrt{2C}}\right) \mathbf{v}_b. \tag{4}$$

We extend $A_1(a,b)$ and consequently $\mathbf{o}_1(a)$ to incorporate reasoning about immediate spatial vicinity of each query a. This is motivated by our observation that adjacent pairs of object queries may provide more important cues for mutually enhancing each other's features than other spatially distant pairs of queries in the image. This is illustrated in Fig. 5. While the main diagonal elements $A_1(a,a)$ have the highest values, non-negligible self-attentions are also estimated for spatially adjacent queries, such as $A_1(a,b)$ for the left remote controller a and the left cat b in the image. We believe that these self-attentions $A_1(a,b)$ between closest neighbors support extracting informative features for each individual query. Furthermore, consider the left remote a and the right remote d that belong to the same class but have $A_1(d,a) \gg A_1(a,d)$. We find that this case often happens for partially occluded instances of the same class. As both remotes are next to the cats, we expect that estimating self-attention of adjacent pairs of queries $A_2((a,b),(d,c))$, called pair self-attention, would help improve learning of the "remote" representation (and thus increase $A_1(a,d)$).

For every target query a, we consider only a pair (a,a'), for efficiency, where a' is the supporting closest query, $a' = \text{argmax}_{a' \in \mathcal{N}_a} \text{IoU}(\mathbf{b}(a), \mathbf{b}(a'))$, and $\mathbf{b}(a)$ is the bounding box of a predicted in the previous decoder output (or initially by the mini-detector). Since the queries are characterized by the positional embedding, it is important to consider their ordering in the pair. We rank the object queries by the L1 distance between their box center in $\mathbf{b}$ and the top-left corner of the image, $L1(\mathbf{b})$, and specify a flip-operator π as

$$\pi a = \begin{cases} a & , \quad L1(\mathbf{b}(a)) \leq L1(\mathbf{b}(a')), \\ a' & , \quad L1(\mathbf{b}(a')) < L1(\mathbf{b}(a)), \end{cases} \tag{5}$$

and similarly $\pi a' = a$ when $L1(\mathbf{b}(a)) \leq L1(\mathbf{b}(a'))$, o.w. $\pi a' = a'$. For query pairs, we define pair self-attention as

$$\begin{aligned} A_2(a,b) &= \text{cat}(\mathbf{q}_{\pi a}, \mathbf{q}_{\pi a'})^\top \text{cat}(\mathbf{k}_{\pi b}, \mathbf{k}_{\pi b'}), \\ &= \mathbf{q}_{\pi a}^\top \mathbf{k}_{\pi b} + \mathbf{q}_{\pi a'}^\top \mathbf{k}_{\pi b'}, \end{aligned} \tag{6}$$

where cat$(\cdot)$ denotes concatenation. From (6), we see that it is critical to account for the right spatial ordering of the pairs, otherwise their respective positional embeddings may be mismatched, as illustrated in Fig. 5.

For every query a, we estimate its output pair self-attention embedding as

$$\mathbf{o}_2(a) = \sum_{b \in \text{queries}} \text{softmax}\left(\frac{A_2(a,b)}{\sqrt{4C}}\right) \mathbf{v}_b. \tag{7}$$

Finally, the individual and pair self-attention outputs are combined after the Add&Norm module as:

$$\mathbf{o}_a = \lambda \cdot \text{norm}(\mathbf{e}_a + \mathbf{o}_1(a)) + (1-\lambda) \cdot \text{norm}(\mathbf{e}_a + \mathbf{o}_2(a)), \tag{8}$$

where λ is a hyper-parameter, norm$(\cdot)$ denotes the layer normalization [1], and $\mathbf{o}_a \in \mathbb{R}^{2C}$.

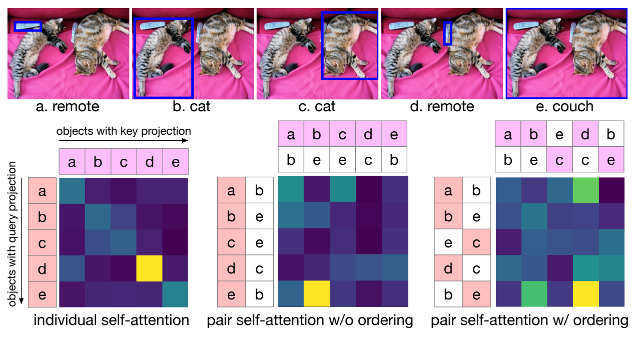

Figure 5. The individual self-attention (bottom left) and pair self-attention maps (bottom middle and right) estimated for five object queries in the image. The **pink** and **purple** indicate the target pair of objects projected into the query and key representation, respectively. The un-colored are support objects of the targets. Pair self-attention without enforcing spatial ordering of (target, support) pairs (bottom middle) gives poor results, because the positional embeddings of the queries may mismatch. This is corrected by enforcing the spatial ordering (bottom right).

3.2.2 Split Cross-attention

As shown in Fig. 4, for every query a, the output of our self-attention $\mathbf{o}_a \in \mathbb{R}^{2C}$ is split into top and bottom halves. The top half represents a classification embedding, whereas the bottom half is a regression embedding, according to the concatenation in (2). These C-dimensional classification and regression embeddings are input separately to the classification and regression branches, where they get linearly projected into the classification query $\mathbf{q}_a^{\text{cls}}$ and the regression query $\mathbf{q}_a^{\text{reg}}$, respectively. The two branches also receive the conditional positional embedding, and share its linear projection into the positional query $\mathbf{q}_a^{\text{pos}}$.

Following C-DETR, we use the concatenations cat$(\mathbf{q}_a^{\text{cls}}, \mathbf{q}_a^{\text{pos}})$ and cat$(\mathbf{q}_a^{\text{reg}}, \mathbf{q}_a^{\text{pos}})$ to represent the object query, and in this way keep the roles of content and position separate in cross-attention. In each branch, these query concatenations are matched with concatenations cat$(\mathbf{k}_n, \mathbf{k}_n^{\text{pos}})$ of the key projection of the image embedding

$\mathbf{f}_n$ obtained from the encoder and the key projection of positional embedding $\mathbf{s}_n$ obtained from the mini-detector as in (1) to compute the following cross-attention outputs:

$$\mathbf{o}_a^{\text{cls}} = \sum_n \text{softmax}\left(\frac{\text{cat}(\mathbf{q}_a^{\text{cls}}, \mathbf{q}_a^{\text{pos}})^\top \text{cat}(\mathbf{k}_n, \mathbf{k}_n^{\text{pos}})}{\sqrt{2C}}\right)\mathbf{v}_n,$$
$$\mathbf{o}_a^{\text{reg}} = \sum_n \text{softmax}\left(\frac{\text{cat}(\mathbf{q}_a^{\text{reg}}, \mathbf{q}_a^{\text{pos}})^\top \text{cat}(\mathbf{k}_n, \mathbf{k}_n^{\text{pos}})}{\sqrt{2C}}\right)\mathbf{v}_n, \tag{9}$$

where $\mathbf{v}_n$ denotes the value projection of the image embedding $\mathbf{f}_n$. In addition to Fig. 1, the supplemental material presents additional visualizations of the classification cross-attention and regression cross-attention that demonstrate advantages of having them separate, as in (9).

$\mathbf{o}_a^{\text{cls}}$ and $\mathbf{o}_a^{\text{reg}}$ are passed to the next decoder stage to serve as the classification and regression embeddings for the next decoder's self-attention. As in C-DETR, we also use the positional embedding of the predicted object center as input to the next decoder stage. In the final decoder stage, $\mathbf{o}_a^{\text{cls}}$ and $\mathbf{o}_a^{\text{reg}}$ are passed through the standard residual link, normalization, and FFN before reaching the classification head and box regression head, respectively. The two heads make the respective set of predictions, which are supervised, as in DETR, using a set-based loss over one-to-one correspondences between the predictions and ground truth.

4. Results

Dataset. For evaluation, we use the MS-COCO 2017 detection dataset [15], and the standard setting: training is performed on 118K training images, and evaluation on 5K images of the val set and 41K images of the test-dev set.

Architecture. Similar to DETR and C-DETR, DESTR has 6 encoder layers, 6 decoder layers, and 8 multi-heads for attention. The mini-detector consists of 3 4-layer FCNs aimed at embedding features from the last encoder layer. The same classification head and bounding box head are used in the mini-detector and all of the decoder layers.

Implementation details. We follow the same training protocol as DETR and C-DETR. The backbone networks are pretrained with ImageNet available on TORCHVISION, and the transformer parameters are initialized with Xavier init [9]. All variants of DESTR are trained with AdamW [18] on an 8-Nvidia-V100s machine. The learning rates for the backbone networks, the mini-detector and the transformer are set to 1e-5, 1e-5 and 1e-4, respectively, and the batch size is set to 16. For the backbone networks with dilation convolution, the batch size is 8. The learning rate decay is set to 0.1, and applied after 40 epochs for the 50-epoch training schedule. 300 objects with the highest classification score are selected for training and testing, if not otherwise specified.

Evaluation metrics. Following the standard COCO evaluation protocols, we report average precision (AP) at 0.50, 0.75 and for small, medium, and large objects, as well as average recall (AR) with maximum 1, 10 and 100 detections, and for small, medium, and large objects.

Loss functions. The same supervision is applied to predictions of the mini-detector and to predictions of the last decoder layer. We use the same loss functions as C-DETR. An optimal bipartite matching between predicted results and the ground truth is obtained by the Hungarian algorithm. Our classification loss is the focal loss, and our box regression loss includes L1 and generalized IoU loss.

4.1. Ablation Studies

Components. Tab. 1 systematically evaluates how each component of DESTR affects performance on COCO-val. The top row shows our strong baseline C-DETR, and the following rows top to bottom gradually extend C-DETR with our contributions. From Tab. 1, as new components are gradually added, performance increases with a reasonably small increase of model complexity. For example, relative to C-DETR, adding the mini-detector or splitting cross-attention, each on its own gives a performance gain, but their combination results in a significant AP increase by 2.0. This supports the claim that our two contributions – the mini-detector and the decoder's split cross-attention – are viable extensions each on their own right, as well as justifies our idea to combine them by initializing the decoder's classification and regression embeddings with the mini-detector's output. Also, adding pair self-attention gives an additional gain of 0.7 in AP. This validates our idea to incorporate prior knowledge by the means of pair self-attention and thus increase DESTR's inductive bias. From Tab. 1, AP drops when we allow the gradients to back-propogate

MiniDet	CASplit	PairAtt	Gflops	# params(M)	AP
			90	44	40.9
✓			96	50	41.5
	✓		95	56	41.4
		✓	93	49	41.6
✓(w/o SG)	✓		101	63	42.1
✓	✓($\mathbf{s}\rightarrow\mathbf{p}$)		101	63	41.8
✓	✓		101	63	42.9
✓	✓	✓(w/o π)	103	69	42.5
✓	✓	✓	104	69	43.6

Table 1. Ablation of components evaluated on COCO-val. Top row is C-DETR, and the following rows to bottom extend C-DETR with our contributions. miniDet denotes the mini-detector, CASplit denotes our splitting of cross-attention, PairAtt means using pair self-attention with $\lambda = 0.5$. The check sign means that the component is included. "w/o SG" means we allow gradient backpropagation from the decoder to the mini-detector. "w/o π" means we do not use the flip-operator π in (6). "$\mathbf{s}\rightarrow\mathbf{p}$" means that we use $\mathbf{p}$ instead of $\mathbf{s}$ in (9).

K (training)	K (testing)	Gflops	AP
300	100	93	41.8
300	200	99	43.2
300	300	104	43.6
300	400	110	43.7
600	300	104	43.5
600	600	122	43.8

Table 2. Performance of DESTR on COCO-val for the varying number of object queries in the decoder K in training and testing.

λ	0.0	0.25	0.5	0.75	1.0
AP	42.4	43.2	43.6	43.0	42.9

Table 3. Results of DESTR on COCO-val for varying the λ weight between individual self-attention and pair self-attention in (8).

from the decoder to the mini-detector, i.e., we observe that the mini-detector overfits without the stop-gradient operation. Finally, AP drops by 1.1 when we do not use the flip-operator π in (6). This validates our claim that the relative position of object queries within a pair should be consistent with queries in another pair for pair self-attention.

The number of object queries K. The mini-detector selects the top K highest classification scoring object proposals, which is also taken as the total number of object queries in the decoder. Tab. 2 shows performance of DESTR on COCO-val for varying K in training and testing. In the following, we will using the same $K = 300$ in both training and testing, since from Tab. 2, this setting gives a good trade-off between AP and complexity, and allows a fair comparison with C-DETR.

Pair attention. Tab. 3 varies the λ weight between individual self-attention and pair self-attention in (8), and shows the best results on COCO-val for $\lambda = 0.5$.

4.2. Comparison with SOTA on COCO

Tab 4 compares DESTR on COCO-val with DETR [2], Deformable DETR (single scale) [30], UP-DETR [4], C-DETR [19], and Anchor DETR [25]. Following DETR and C-DETR, we report AP values for 4 backbone networks: ResNet-50, ResNet-101, ResNet50-DC, and ResNet101-DC, where "DC" denotes using dilated C5 features, while the others use original C5 features. From Tab 4, DESTR outperforms the strong baseline C-DETR, in all AP metrics, for all four backbones. While Deformable DETR has been design primarily for the multi-scale features, DESTR outperforms its single-scale variant Deformable DETR-SS. DESTR has a larger performance gain over SOTA with ResNet-50 than with ResNet50-DC and ResNet101-DC. Based on the DETR GitHub discussions, we believe that this is because of the larger batch sizes used by SOTA for ResNet50-DC and ResNet101-DC, while our hardware limits our batch size to be only 1image/GPU $\times$ 8GPUs. Tab 4 shows inferior results of TSP-FCOS-R50 and TSP-RCNN-R50 [23] to ours for the same ResNet-50 backbone, although they use multi-scale features.

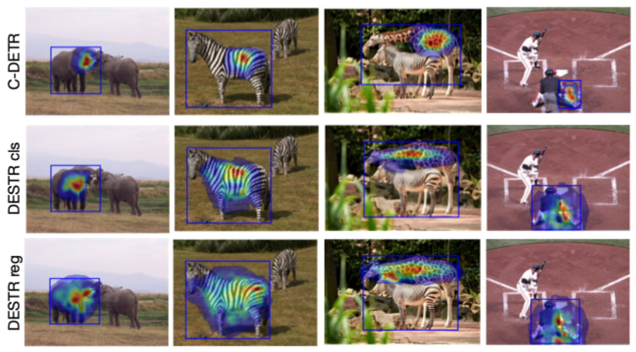

Figure 6. Out of 8 attention multi-heads, this figure visualizes the cross-attention map for the head focusing on the object's center while the other maps are given in the supplement. For clarity, we show square-root values of attention. C-DETR's cross-attention for the object's center focuses on a relatively small region, whereas DESTR's classification and regression cross-attention maps have a broader spatial support including the object's immediate surrounding, as intended by our pair attention.

Tab. 5 compares DESTR on COCO-test-dev with the following SOTA detectors: FCOS [24], ATSS [27], Deform-DETR [30], and C-DETR [19]. DESTR achieves the highest AP and AR scores among the approaches that use single-scale features, for all four backbones. DESTR also outperforms CNN-based FCOS [24] and ATSS [27] for the same backbones, although they use multi-scale features.

Fig. 6 shows cross-attention maps estimated for example images from COCO. Out of 8 attention multi-heads, the figure visualizes the cross-attention map for the head focusing on the object's center while the other maps are given in the supplement. DESTR's classification and regression cross-attention maps differ, as expected, since they focus on different visual cues for their respective tasks. Also, both classification and regression cross-attention maps of DESTR highlight a larger spatial support than C-DETR's attention. This suggests that our cross-attention additionally seeks important cues from the object's immediate vicinity.

5. Conclusion

We have specified DESTR that extends the recent family of Transformer-based object detectors with three contributions: 1) cross-attention is split into the independent classification and regression branches so attention could optimally focus on relevant visual cues for the respective tasks; 2) a mini-detector is used to learn and initialize both content and positional embeddings of the decoder, and enable hav-

Model	#epochs	GFLOPs	#params (M)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR-R50 [2]	500	86	41	42.0	62.4	44.2	20.5	45.8	61.1
Deform-DETR-R50-SS [30]	50	**78**	**34**	39.4	59.6	42.3	20.6	43.0	55.5
UP-DETR-R50 [4]	150	86	41	40.5	60.8	42.6	19.0	44.4	60.0
UP-DETR-R50 [4]	300	86	41	42.8	63.0	45.3	20.8	47.1	61.7
C-DETR-R50 [19]	50	90	44	40.9	61.8	43.3	20.8	44.6	59.2
Anchor DETR-R50 [25]	50	–	–	42.1	63.1	44.9	22.3	46.2	60.0
DESTR-R50	50	104	69	**43.6**	**64.7**	**46.5**	**23.6**	**47.5**	**62.1**
DETR-DC5-R50 [2]	500	187	41	43.3	63.1	45.9	22.5	47.3	61.1
Deform-DETR-DC5-R50-SS [30]	50	**128**	**34**	41.5	61.8	44.9	24.1	45.3	56.0
C-DETR-DC5-R50 [19]	50	195	44	43.8	64.4	46.7	24.0	47.6	60.7
Anchor DETR-DC5-R50 [25]	50	151	–	44.2	64.7	47.5	24.7	48.2	60.6
DESTR-DC5-R50	50	232	69	**45.3**	**65.7**	**48.3**	**27.3**	**48.8**	**62.4**
DETR-R101 [2]	500	**152**	**60**	43.5	63.8	46.4	21.9	48.0	61.8
C-DETR-R101 [19]	50	156	63	42.8	63.7	46.0	21.7	46.6	60.9
Anchor DETR-R101 [25]	50	–	–	43.5	64.3	46.6	23.2	47.7	61.4
DESTR-R101	50	171	88	**44.6**	**65.4**	**47.8**	**24.1**	**48.7**	**63.8**
DETR-DC5-R101 [2]	500	**253**	**60**	44.9	64.7	47.7	23.7	49.5	62.3
C-DETR-DC5-R101 [19]	50	262	63	45.0	65.5	48.4	26.1	48.9	60.7
Anchor DETR-DC5-R101 [25]	50	–	–	45.1	65.7	48.8	25.8	49.4	61.6
DESTR-DC5-R101	50	299	88	**46.4**	**67.1**	**50.1**	**28.2**	**50.3**	**63.7**
TSP-FCOS-R50 [23]	36	189	-	43.1	62.3	47.0	26.6	46.8	55.9
TSP-RCNN-R50 [23]	36	188	-	43.8	63.3	48.3	28.6	46.9	55.7

Table 4. Comparison with other DETR variants on COCO-val for 4 backbone networks: ResNet-50 (R50), ResNet-101 (R101), ResNet50-DC, and ResNet101-DC, where "DC" denotes using dilated C5 features. Note that TSP-FCOS-R50 and TSP-RCNN-R50 [23] in the bottom block use multiscale features, so the double horizontal separation indicates that a direct comparison is not fair to us.

Model	#epochs	feature scale	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_{m1}	AR_{m10}	AR_{m100}	AR_S	AR_M	AR_L
C-DETR-R50 [19]*	50	ss	41.1	62.3	43.8	19.2	44.3	57.4	33.6	54.4	58.6	31.6	63.9	82.3
DESTR-R50	50	ss	**43.3**	**64.5**	**46.3**	**21.8**	**46.5**	**60.3**	**35.0**	**56.8**	**60.9**	**35.1**	**66.3**	**83.4**
C-DETR-DC5-R50 [19]*	50	ss	43.7	64.6	46.8	22.7	47.0	59.1	35.0	57.5	61.7	36.3	66.9	83.8
DESTR-DC5-R50	50	ss	**45.3**	**66.0**	**48.9**	**24.9**	**48.3**	**60.0**	**35.9**	**59.3**	**63.8**	**40.2**	**68.3**	**84.3**
C-DETR-R101 [19]*	50	ss	43.1	64.5	46.1	21.4	46.4	59.9	34.7	56.3	60.3	34.0	65.6	83.6
DESTR-R101	50	ss	**44.9**	**66.1**	**48.0**	**22.8**	**48.3**	**61.5**	**35.7**	**58.2**	**62.3**	**36.6**	**67.9**	**84.6**
C-DETR-DC5-R101 [19]*	50	ss	45.4	66.3	48.9	24.3	48.8	61.5	36.1	58.8	63.0	38.1	68.3	84.7
DESTR-DC5-R101	50	ss	**46.8**	**67.6**	**50.5**	**25.8**	**50.1**	**62.3**	**36.8**	**60.3**	**64.8**	**41.2**	**69.9**	**85.2**
FCOS-R101 [24]	24	ms	41.5	60.7	45.0	24.0	44.2	51.3	-					
ATSS-R101 [27]	24	ms	43.6	62.1	47.4	26.1	47.0	53.6	-					
ATSS-DC-R101 [27]	24	ms	46.3	64.7	50.4	27.7	49.8	58.4	-					
Deform-DETR-R50 [30]	50	ms	46.9	66.4	50.8	27.7	49.7	59.9	-					
Deform-DETR-R101 [30]	50	ms	48.7	68.1	52.9	29.1	51.5	62.0	-					

Table 5. Comparison with SOTA on COCO-test-dev. "-" means the results have not been reported. "*" means we test the models with weights provided on the authors' official GitHub. "ms" and "ss" denote multi-scale and single-scale features used by the model. Note that the SOTA approaches in the bottom block use multi-scale features, so the double horizontal separation indicates that a direct comparison is not fair to us.

ing an adaptive inferred number of object queries; 3) self-attention is estimated over pairs of adjacent object queries so attention of each query could be enhanced by the immediate spatial context of the other query in the pair. An extensive ablation study reported in this paper has demonstrated performance gains for each of these contributions and combinations thereof, relative to the strong baseline of C-DETR. Our experiments have shown that, under the standard evaluation settings on the COCO-val and COCO-test-dev datasets, DESTR outperforms the SOTA Transformer detectors which use single-scale features in both AP and AR with comparable model complexity. For limitation, one could point out that the mini-detector breaks the well-streamlined architecture of DETR; however, the mini-detector is used only for initialization of the content queries, and DESTR's final detections are not explicitly constrained by the mini-detector's predictions. As any system for object detection, ours could be misused for malicious human monitoring and violations of privacy.

6. Acknowledgement

DARPA MCS Award N66001-19-2-4035

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 1, 2, 3, 4, 7, 8

[3] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *ICLR*, 2020. 1

[4] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1601–1610, 2021. 7, 8

[5] Zihang Dai, Hanxiao Liu, Quoc V. Le, and Mingxing Tan. CoAtNet: Marrying convolution and attention for all data sizes. *CoRR*, 2021. 1

[6] 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. 3

[7] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019. 3

[8] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. *arXiv preprint arXiv:2101.07448*, 2021. 3

[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 6

[10] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015. 3

[11] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018. 3

[12] Hei Law, Yun Teng, Olga Russakovsky, and Jia Deng. Cornernet-lite: Efficient keypoint based object detection. *arXiv preprint arXiv:1904.08900*, 2019. 3

[13] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6054–6063, 2019. 3

[14] 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. 3

[15] 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, 2, 6

[16] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 3

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 3

[18] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *ICLR*, 2018. 6

[19] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional DETR for fast training convergence. *arXiv preprint arXiv:2108.06152*, 2021. 1, 2, 3, 7, 8

[20] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10213–10224, 2021. 1

[21] Han Qiu, Yuchen Ma, Zeming Li, Songtao Liu, and Jian Sun. Borderdet: Border feature for dense object detection. *arXiv preprint arXiv:2007.11056*, 2020. 3

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. 1

[23] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris M Kitani. Rethinking transformer-based set prediction for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3611–3620, 2021. 3, 7, 8

[24] 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, 3, 4, 7, 8

[25] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. *arXiv preprint arXiv:2109.07107*, 2021. 1, 3, 7, 8

[26] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520, 2016. 3

[27] 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, 7, 8

[28] Chenchen Zhu, Fangyi Chen, Zhiqiang Shen, and Marios Savvides. Soft anchor-point object detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 91–107. Springer, 2020. 3

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

[30] 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, 7, 8