D^{2}ETR: DECODER-ONLY DETR WITH COMPUTATIONALLY EFFICIENT CROSS-SCALE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

DETR is the first fully end-to-end detector that predicts a final set of predictions without post-processing. However, it suffers from problems such as low performance and slow convergence. A series of works aim to tackle these issues in different ways, but the computational cost is yet expensive due to the sophisticated encoder-decoder architecture. To alleviate this issue, we propose a decoder-only detector called D^{2}ETR. In the absence of encoder, the decoder directly attends to the fine-fused feature maps generated by the Transformer backbone with a novel computationally efficient cross-scale attention module. D^{2}ETR demonstrates low computational complexity and high detection accuracy in evaluations on the COCO benchmark, outperforming DETR and its variants.

1 INTRODUCTION

Object detection is a computer vision task to predict category labels and bounding box locations for all objects of interest in an image. Modern detectors (Redmon & Farhadi, 2018; Tian et al., 2019; He et al., 2017) treat this set prediction task as regression and classification problems. They optimize the predicted set with many hand-crafted components such as anchor generation, training target assignment rule, and non-maximum suppression (NMS). These components complicate the pipeline and the detectors are not end-to-end.

DETR (Carion et al., 2020) proposed to build the first fully end-to-end detector, featuring an encoder-decoder Transformer architecture that predicts a final set of bounding boxes and category labels without any well-designed anchor, heuristic assignment rule, and post-processing. The fancy design of DETR has been receiving a lot of research attention. However, DETR suffers from many problems such as slow training convergence, low performance on small objects, and high computational complexity. A bunch of works (Zheng et al., 2020; Sun et al., 2020; Zhu et al., 2020; Yao et al., 2021; Meng et al., 2021; Gao et al., 2021) aim to handle these critical issues. Many efforts have been made to efficient cross-attention to accelerate the training convergence. Multi-scale feature maps are also used to improve the accuracy of small objects. Though the works above have made some progress, the problem of high computational complexity is left untouched.

In the field of Natural Language Processing (NLP), the OpenAI GPT series (Radford et al., 2019; Brown et al., 2020) adopt a decoder-only Transformer but show impressive ability on text generation. It implies that a rigorous encoder-decoder is not necessary in language modeling, which motivates us to rethink the DETR architecture. In DETR, the decoder is the key for locating with object queries, while the encoder, which produces self-attentive features, only plays as an assistant for the subsequent decoder. To validate it, we re-examined the overall impact of encoder by training Deformable DETR w/o encoder. We found that the encoder module brings 4.9 (+11%) AP improvement but costs very large portion of computation, about 85 (+49%) GFLOPs. Such a low computional efficiency of encoder prompts us to explore the possibility of removing the Transformer encoder by integrating feature extraction and self-attention based fusion functions within a single backbone, creating a simpler decoder-only detection pipeline.

In this paper, we propose *Decoder-only DEtection TRansformer* (D^{2}ETR), which achieves fast convergency, high performance, and low computational cost with an easy architecture. To introduce the interaction among features at different locations and scales, our method leverages the advantage of Transformer backbones that providing a global receptive field of intra-scale, plus a novel module

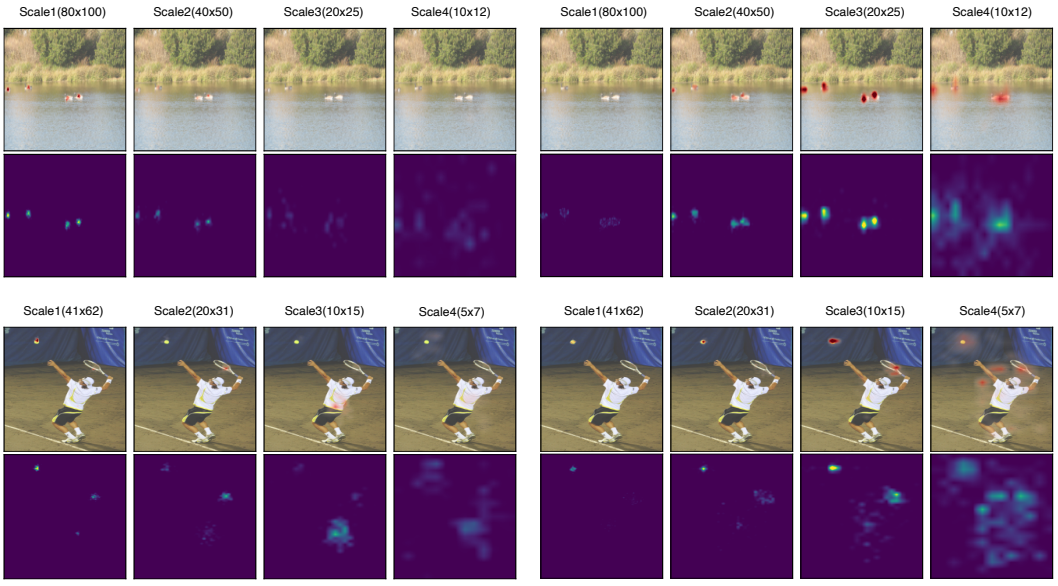


Figure 1: Visualization of the cross-attention on different scales. (a) Deformable DETR. (b) Deformable D^2ETR. Check Appendix A.5 for more visualizations.

Computationally Efficient Cross-scale Attention (CECA) that performing sparse feature interaction of cross-scale via attention mechanism. By organizing the cross-attantion in which high-level feature maps as query and the remained lower-level feature as key-value pairs, CECA captures the low-level visual feature helpful for fine-grained location, but prevents the computation explosion when locating on low-level feature maps directly. With such a design, the decoder can directly attend to the fine-fused feature maps generated by our backbone, without the aid of an encoder or other fusion block to introduce more feature interactions.

The CECA module can easily replace the encoder and go with any type of decoder to form an end-to-end detector flexiblely. We cooperate it with a decoder of standard multi-head attention and a decoder of deformable attention, resulting in a vanilla D^2ETR and a Deformable D^2ETR. Futhermore, we introduce two auxiliary losses: (i) *token labeling loss*, which helps improve the accuracy by enhancing feature expression capabilities. (ii) *IoU-awareness loss*, helps improve the localization accuracy by adding constrains the predicted bounding boxes.

Figure 1 visualizes the cross-attention on different scales. Deformable DETR focuses more on low-level scales of high resolution, which requires carefully fine-fused and thus computationally inefficient. Our method focuses more on the high-level scales, which have aggregated information from all previous scales in a coarse-to-fine manner. In evaluation on the COCO 2017 detection benchmark (Lin et al., 2014), our proposed methods achieve competitive performance at low computational complexity, e.g., 43.2 AP and 82 GFLOPs for D^2ETR, 50.0 AP and 93 GFLOPs for Deformable D^2ETR.

2 Related Work

End-to-end Object Detections. Conventional one-stage (Lin et al., 2017b; Redmon & Farhadi, 2018; Tian et al., 2019) and two-stage detectors (Ren et al., 2015; He et al., 2017; Cai & Vasconcelos, 2018) rely on anchor boxes or anchor centers. Due to the dense anchors, an intersection-over-unit (IoU) based one-to-many assignment rule is usually applied to training, and a non-maximum suppression (NMS) is used to resolve duplications during inference. Unlike aforementioned detectors, end-to-end detectors eliminate the need for post-processing by learning to resolve the duplications. DETR (Carion et al., 2020) initiatively applies an encoder-decoder Transformer architecture to a

CNN backbone and builds an end-to-end detector. However, DETR has some issues such as slow training convergence, limited feature spatial resolution, and low performance on small objects.

Many variants have been proposed to address these critical issues. ACT (Zheng et al., 2020) adaptively clusters similar query elements together. Deformable DETR (Zhu et al., 2020) uses multi-scale feature maps to help detect small objects. It introduces the deformable attention mechanism that only focuses on a small fixed set of sampling points predicted from the feature of query elements. This modification mitigates the issues of convergence and feature spatial. Conditional DETR (Meng et al., 2021) presents a conditional cross-attention mechanism. A spatial embedding is predicted from each output of the previous decoder layer, and then fed to next cross-attention to make the content query localize distinct regions. SMCA (Gao et al., 2021) conducts location-constrained object regression to accelerate the convergence by forcing co-attention responses to be high near initially estimated bounding box locations. YOLOS (Fang et al., 2021) argues that object detection tasks can be completed in a pure sequence-to-sequence manner. Similarly, Pix2Seq (Chen et al., 2021) treats object detection as a language modeling task conditioned on pixel inputs.

Our proposed D^{2}ETR focuses on removing the whole encoder at minimum cost to simplify the pipeline and ease the high computation consumption.

Multi-scale Feature Fusion. A line of works have been done on feature fusion and prove that a good spatial feature fusing scheme is necessary for conventional object detection, especially when detecting small objects. FPN (Lin et al., 2017a) combines two adjacent feature maps and builds a top-down feature pyramid. PAN (Liu et al., 2018) adds an extra bottom-up path augmentation. NAS-FPN (Ghiasi et al., 2019) utilizes neural architecture search to find an optimal feature pyramid structure. In the field of end-to-end object detection, the encoder fuses feature maps via attention mechanism, playing a similar role as the feature pyramid network. Sun et al. (2020) replaces the decoder with a detection head and directly uses the outputs of encoder for object prediction, which means the encoder is good at extract context features. Yao et al. (2021) conducts an experiment on the effect of different numbers of encoder layers and decoder layers and claims that detectors based on encoder-decoder architecture are more sensitive to the number of decoder layers, which implies the encoder is less efficient.

These observations motivates us to seek a more economical way to exchange information on multi-scale feature maps. Our computational efficient cross-scale attention allows the backbone to generate fine-fused feature maps without the need for an encoder.

3 Revisiting Encoder-decoder Architecture

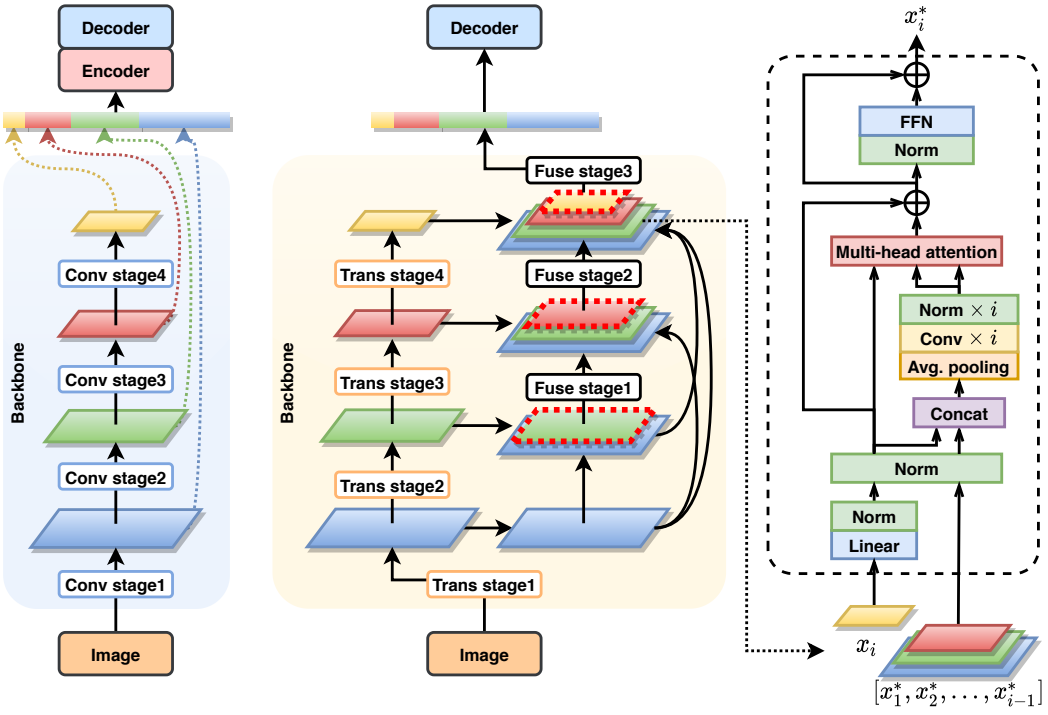
The DETR family is based on the encoder-decoder Transformer architecture. Figure 2(a) show the three major parts of an end-to-end detector: backbone, encoder, and decoder. The backbone extracts feature maps $x \in \mathbb{R}^{C\times H\times W}$ of the input image. Then the pixels in x attend to each other with Transformer encoder layer, which consists of a Self-Attention (SA) and a Feed-Forward Network (FFN). The standard self-attention module (Vaswani et al., 2017) computes attention weights based on the similarity between query-key pairs, and then compute a weighted sum over all key contents. We can fomulate the encoder as:

$$\text{SA}(x_q, x_k, x_v) = \text{softmax}\left(\frac{x_q W_q (x_k W_k)^T}{\sqrt{C}}\right) x_v W_v \tag{1}$$

$$\text{FFN}(x) = \sigma(xW_1 + b_1)W_2 + b_2 \tag{2}$$

where $x_q, x_k, x_v = \text{Flatten}(x)$ in encoder, $\text{Flatten}(\cdot)$ is the operation that flatten x along spatial dimensions, and $\sigma(\cdot)$ is the non-linear activation. The self-attentive features are subsequently fed into decoder. The decoder has similar structures with an extra cross-attention module, which modifies x_q as $o_q \in \mathbb{R}^{N\times C}$ in Eq 1. o_q is the learned object queries.

We give a complexity analysis of encoder and decoder. For encoder whose input dimension is $H \times W \times C$, the complexity can be calculated as $\mathcal{O}(H^2W^2C + HWC^2)$. Compared with encoder, the decoder adopts N object queries for cross-attention, which has the complexity of $\mathcal{O}(NHWC + NC^2)$. Generally, a relatively small number of object queries is sufficient for localization in end-to-end methods. While the amount of elements in feature map, which always a large resolution,



(a) DETR series (b) D²ETR (c) i-th fusing stage

Figure 2: Illustration of (a) original DETR series, (b) D²ETR, (c) one layer of the i-th fusing stage.

is much greater than N. That is to say, the encoder have much larger computational complexity compared to the decoder, especially when the size of input feature map is large.

4 DECODER-ONLY DETR

In essence, the encoder is a combination of intra-scale and cross-scale feature interaction. The self-attention mechanism of Transformer naturally introduces intra-scale interaction to separate feature maps. It motivates us to fill the missing cross-scale interaction and build a powerful Transformer backbone to generate fine-fused features, and further take over the inefficient encoder.

To perform feature interaction across feature maps of all scales, a naive design is to apply a dense connection to the model. Given x_i as the original i-th feature map, x_i^j as the i-th feature map after j times of fusions, $\mathbf{H}_i$ as the Transformer block of i-th stage. At i-th stage, the feature map x_i can be formulated as $\mathbf{H}_i([x_1, x_2, ..., x_{i-1}])$, where $[\cdot]$ denotes a concatenation of elements. Scales of each stage are linearly projected and spatial-wisely concatenated to the next stage to generate new scale, and do further cross-scale feature fusion. The fusing function is additional self-attention, denoted as $\text{SA}(x_q, x_k, x_v)$, where $x_q = x_k = x_v = [x_1^{i-1}, x_2^{i-2}, ..., x_i]$. The final output of the backbone is $[x_1^S, x_2^{S-1}, ..., x_S^1]$, given the last S feature maps as the input of decoder. This dense architecture is a combination of feature extraction and fusion. However, it makes no difference from the original encoder because the low-level scales are of high-resolution and take part in almost all self-attention operations, leading to expensive computation and a waste of cross-attention from decoder.

4.1 D²ETR ARCHITECTURE

Inspired by the dense connected Transformer in the previous section, we present a Computationally Efficient Cross-scale Attention (CECA) and build a Decoder-only DETR (D²ETR), illustrated in Figure 2(b). The architecture consists of two main components: a Transformer backbone and a Transformer decoder. The backbone is the core of encoder-free. It contains two parallel streams,

one for intra-scale interaction and another for cross-scale interaction. Transformers that have linear computational complexity w.r.t. image size are preferred in object detection. By default, we borrow the idea of Pyramid Vision Transformer (PVT) (Wang et al., 2021) to build our backbone. We will show that D^{2}ETR can cooperate with different Transformers in the ablation. A decoder can learn to generate non-duplicated detections, which is the key that makes the detector end-to-end. Our D^{2}ETR can equip any type of Transformer decoder without encoder.

Computationally Efficient Cross-scale Attention The idea of dense fusion is promising for integrating feature fusion to the backbone. As mentioned above, the main problem is the large number of query elements in self-attention. To address this, we decouple the intra-scale and cross-scale interaction and fuse the feature map in a sparse manner. In Figure 2(b), the backbone is divided into four Transformer stages that produce feature maps of different scales. The scale of the output feature maps progressively shrinks. All stages have a similar architecture which depends on the basic block of the chosen Transformer. Following the PVT implementation, the stage consists of an overlapping patch embedding and multiple successive Transformer layers built by spatial reduction self-attention and convolutional feed-forward module. All the feature maps are a global content aggregation within their own scale.

The flow in parallel is the fusing stages. The fusing stage is designed for cross-scale feature fusion. Each desired scale enters a fusing stage as the query element (the feature map with red dashed line) and all existing fused scales are densely connected to the same fusing stage as the key elements. A modified spatial reduction operation is applied to key elements to alleviate the computational cost. The query scale can eventually aggregate visual features at spatial locations of all previous scales. The proposed CECA can be formulated as:

$$x_i = \mathbf{H}_i(x_{i-1}) \tag{3}$$

$$x_i^* = \text{SA}(x_q, x_k, x_v) \tag{4}$$

$$x_q = x_i, \quad x_k = x_v = [x_1^*, x_2^*, ..., x_{i-1}^*, x_i] \tag{5}$$

where x_i^* stands for the the fused version of feature map x_i. Given the last S feature maps as the input of decoder, the final output of CECA would be $[x_1^*, x_2^*, ..., x_S^*]$.

Figure 2(c) details one layer of i-th fusing stage. It is composed of three parts: a linear spatial reduction, a multi-head self-attention, and a feed-forward layer. x_i stands for the feature map of the query scale. $[x_1^*, x_2^*, ..., x_{i-1}^*]$ stands for the feature maps of the key scales from preceding dense connections. To prevent the query scale from losing its own high-level characteristics during information exchange, we project it with the proper channel number and concatenate to the key scales. To reduce the computational cost, the key elements are fed to linear spatial reduction, which is an adaptive average pooling layer followed by i separate 1×1 convolutional layers and norm layers for feature maps of each scale. In the multi-head attention module, x_i as the query interacts with $[x_1^*, x_2^*, ..., x_{i-1}^*, x_i]$ as the keys to extract context information. The FFN we consider is the feed-forward with an additional depth-wise convolution. Same as the normal stage, this layer is repeated multiple times. Details of fusing stage can be found in Appendix A.1.

Given h, w as the height and width of the last feature map x_S at the last stage. For the naive dense fusion, the complexity to compute self-attentionis $\mathcal{O}(4^S ShwP^2C)$ where P is the adaptive pooling size. The complexity is dominated by the S, in other words, sizes of the low-level feature maps. Our CECA enjoys the cost-effective fusion emphasizing high-level feature maps. As a result, the complexity is reduced to $\mathcal{O}(ShwP^2C)$. More details can be found in Appendix A.2

By applying Transformer to the backbone, we introduce intra-scale interaction to pixels on separate feature maps. By adding extra fusing stages, we introduce cross-scale interaction to pixels on multi-scale feature maps. D^{2}ETR structure fusing stages with single query scale and densely connected key elements. In this way, the number of query elements is greatly reduced, which allows us to use deeper fusing stages. Meanwhile, the subtle visual feature of low-level scales is kept to the Transformer decoder which helps to refine predictions, especially for small objects.

Vs. Transformer Encoder. In end-to-end object detectors, the Transformer encoder servers as an independent feature refiner to fuse features *after* the backbone finished generating the very last feature map. Spatial locations at feature maps of all scales are exchanging information as equal

individuals. In contrast, our proposed CECA is affiliated to the backbone, which allows the backbone to generate fine-fused feature maps. The feature interaction is split into intra-scale and cross-scale, providing by Transformer stages and fusing stages, respectively. Pixels of different scales are not symmetrical. Note that our backbone works as a whole so it is possible to get a better initialization in downstream tasks if CECA joins the pre-train. Moreover, because the depth of fusing stages is usually less than Transformer stages, we can further execute $(i+1)$-th stage and i-th fusing stage simultaneously to hide the latency introduced by the auxiliary structure. These optimizations are impossible to the conventional encoder-decoder architecture.

4.2 Loss Function.

Our proposed CECA backbone can generate fine-fused features and go with any type of decoder to form an end-to-end detector. The vanilla DETR decoder applies the standard multi-head attention and uses only one feature map due to the high computational cost. The Deformable DETR decoder employs deformable attention to extract context information from the multi-scale feature maps. To validate the flexibility, we cooperate with the above two decoders and build D^{2}ETR and Deformable D^{2}ETR, which use single-scale and multi-scale feature maps, respectively. Futhermore, we introduce two auxiliary losses, token labeling loss and IoU-awareness loss, the total loss function can be formulated as:

$$\mathcal{L}_{total} = \mathcal{L}_{cls} + \mathcal{L}_{bbox} + \mathcal{L}_{awr} + \mathcal{L}_{token} \tag{6}$$

where $\mathcal{L}_{cls}$ denotes the classification loss, $\mathcal{L}_{bbox}$ denotes the regression loss, $\mathcal{L}_{awr}$ is the loss of awareness branch, $\mathcal{L}_{token}$ is the loss of token labeling.

IoU-awareness. Predicted bounding boxes of few overlaps with target objects tend to be of low quality. To alleviate the mismatch between localization quality and detection confidence, we adopt the *IoU-awareness* loss proposed by Wu et al. (2020). Specifically, a new branch is added to the top of all decoder layers, predicts the IoU between the predicted bounding box and the target one, and then integrated with the classification score to suppress low-quality predictions. We formulate the awareness loss term as:

$$\mathcal{L}_{awr} = \frac{1}{B}\sum_{i=1}^{B} \text{BCE}\left(\text{FFN}(\hat{y}_i), \text{IoU}(b_i, \hat{b}_i)\right) \tag{7}$$

where B is the number of bounding boxes, $\hat{y}_i$ is the output corresponding to i-th object query, and passes through FFN and yields a predicted IoU. $b_i, \hat{b}_i$ represents the target and predicted bounding box, respectively. The higher predicted IoU value, the higher possibility for the corresponding bounding box to capture a real target. During inference, the output of classification branch is multiplied by the predicted IoU to filter low-quality results. Check Appendix A.3 for more details.

Token Labeling. *Token labeling* (Jiang et al., 2021) is a novel training objective for Transformers that performs patch-wise classifications in image classification task. We adopt token labeling to the training phase of object detection. Specifically, we utilize mask annotations to supervise and interpolate it to align with the resolution of feature map. Each pixel is assigned with a soft token label and performs multiclass classification, which encourages the backbone to extract more strength features. The loss term of token labeling can be defined as:

$$\mathcal{L}_{token} = \frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{N}\sum_{p,q} \text{Focal}\left(\text{FFN}(x_j[p,q]), t_j[p,q]\right) \tag{8}$$

$$t_j = \text{Interpolate}(M, \text{SizeOf}(x_j)) \tag{9}$$

where $x_j[p,q]$ denotes the features at position (p,q) of j-th feature map from the backbone, t_j is the corresponding target soft token label, generated by interpolating the 0-1 mask annotation M to the same size of x_j. See Appendix A.4 for more details.

Table 1: Comparison with modern end-to-end detectors on COCO 2017 validation set.

Method	Backbone	Epoch	GFLOPs	Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR	R50	500	86	41	42.0	62.4	44.2	20.5	45.8	61.1
DETR	PVT2	108	79	40	42.4	63.4	44.9	19.8	45.8	63.0
Conditional DETR	R50	108	90	44	43.0	64.0	45.7	22.7	46.7	61.5
DETR	DC5-R50	500	187	41	43.3	63.1	45.9	22.5	47.3	61.1
Deformable DETR	R50	50	173	40	44.5	63.6	48.7	27.1	47.6	59.6
Deformable DETR	PVT2	50	163	34	48.3	67.6	52.9	30.5	51.6	63.8
TSP-RCNN	R50	36	188	-	43.8	63.6	48.3	28.6	46.9	55.7
TSP-FCOS	R50	36	189	-	43.1	62.3	47.0	26.6	46.8	55.9
Efficient DETR	R50	36	159	32	44.2	62.2	48.0	28.4	47.5	56.6
SMCA	R50	50	152	40	43.7	63.6	47.2	24.2	47.0	60.4
D^2ETR	PVT2	50	82	35	43.2	62.9	46.2	22.0	48.5	62.4
Deformable D^2ETR	PVT2	50	93	40	50.0	67.9	54.1	31.7	53.4	66.7

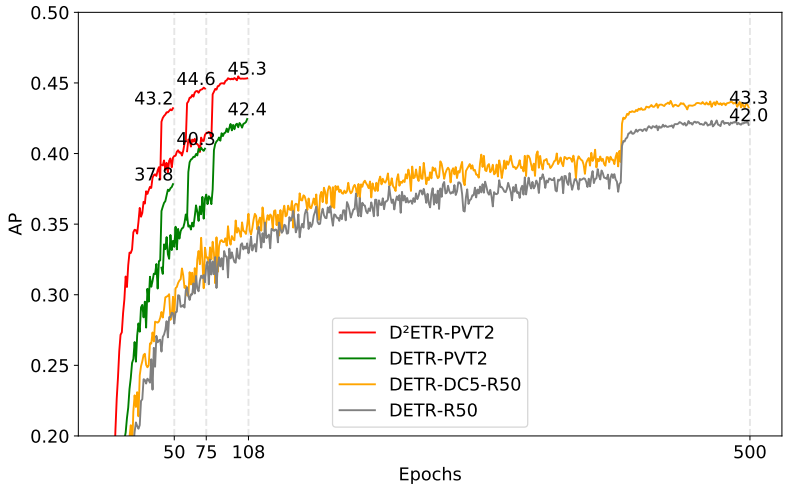


Figure 3: Convergence curves. Results of DETR trained for 500 epochs are from DETR's official Github repository.

5 Experiments

Dataset. We perform experiments on the COCO 2017 detection dataset (Lin et al., 2014). We train our models on the training set and evaluate on the validation set.

Implementation Details. We follow the implementation of PVT-v2-B2-linear (Wang et al., 2021). Model pre-trained in ImageNet-1K (Deng et al., 2009) is used to initialize the Transformer stages of our backbone. All the fusing stages have a depth of 3, a channel number of 256, an expansion ratio of 4. Other hyperparameters are kept the same with its nearest normal stage. We follow the Deformable DETR training settings. The decoder has 6 layers, and the number of object queries is 300. For D^2ETR, we use a single-scale feature map of stride 32. For Deformable D^2ETR, two-stage mode are enabled and we use multi-scale feature maps of strides 8, 16, 32, 64, where the last feature map is obtained via a 3×3 convolution layer. The optimizer is an AdamW (Loshchilov & Hutter, 2017) with base learning rate of 2×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay of 1×10^{-4}. By default, we train our model for 50 epochs with a batch size of 32, and the learning rate is dropped by a factor of 0.1 at epoch 40. Random crop and resize are used for data augmentation with the largest side length set as 1333. Training and evaluation run on NVIDIA 2080Ti.

Table 2: Comparison with DETR and Deformable DETR on COCO 2017 validation set.

Method	Backbone	Decoder-only?	Multi-scale?	Epoch	GFLOPs	Params	AP	AP_{50}	AP_{75}
DETR	R50			50	86	41	34.9	55.5	36.0
DETR	R50	✓		50	76	33	29.0	48.9	29.4
D^{2}ETR	R50	✓		50	88	37	38.8	58.1	41.1
DETR	PVT2			50	79	40	37.8	59.1	39.6
DETR	PVT2	✓		50	69	32	33.2	54.5	34.3
D^{2}ETR	PVT2	✓		50	82	35	43.2	62.9	46.2
Deformable DETR	R50		✓	50	173	40	44.5	63.6	48.7
Deformable DETR	R50	✓	✓	50	88	35	39.6	59.9	42.8
Deformable DETR	PVT2		✓	50	163	34	48.3	67.6	52.9
Deformable DETR	PVT2	✓	✓	50	78	30	43.5	64.2	47.3
Deformable D^{2}ETR	PVT2	✓	✓	50	93	40	50.0	67.9	54.1

5.1 RESULTS

As shown in Table 1, our method is compared with modern end-to-end detectors. We report GFLOPs for first 100 images of the validation set.

Main Results. We observe that DETR requires 500 training epochs to converge and the performance is relatively low. Conditional DETR boosts the convergency but the accuracy increment is less than obvious. Variants such as Deformable DETR, Efficient DETR, and SMCA reach better performances, yet the computational complexity remains at a high level. The proposed D^{2}ETR achieves a competitive 43.2 AP and the low computational complexity of 82 GFLOPs with $10\times$ fewer training epochs w.r.t. original DETR. Detailed convergence curves can be found in Figure 3, we run 3 different training epochs 50, 75, 108 with learning rate drops at epoch 40, 60, 80, respectively. The proposed Deformable D^{2}ETR surpasses all end-to-end detector baselines with the highest detection accuracy of 50.0 AP and a quite low computational complexity of 93 GFLOPs.

Effect of Backbone. Table 2 compares the proposed methods with DETR and Deformable DETR after the same training epochs of 50. Transformer is not only a better feature extractor but also an indispensable instrument for us to conduct feature fusion in separate scales. To understand the influence of Transformer backbone, we try DETR and Deformable DETR with PVT2. It greatly improves the performance, but the computational cost doesn't change too much. Take Deformable DETR-PVT2 as an example, it still demands a lot of effort (163 GFLOPs) on the encoder to refine the low-level feature maps. In contrast, the proposed Deformable D^{2}ETR-PVT2 make use of high-level feature maps during cross-scale feature exchange, which delivering noticeable lower computational cost (93 GFLOPs). We also try to cooperate D^{2}ETR with the CNN backbone ResNet50. It enjoys the benefits of cross-scale attention and improves the DETR-R50 baseline with 3.9 AP, yet lacking adequate intra-scale attention and the performance is 4.4 AP lower than the proposed method. This reveals the necessity of a Transformer backbone.

Effect of Encoder. Unlike other works using distinct attention schemes, we focus on removing the whole encoder at minimum cost to simplify the pipeline. To analyze the impact of the absence of encoder, we run DETR and Deformable DETR in the decoder-only mode, as shown in Table 2. We can see that the absence of encoder does alleviate the model complexity (up to 85 GFLOPs reduction), yet leading to a huge slope of the accuracy as well (up to 5.9 AP reduction). Our proposed approach prevents degradation with the computational efficient cross-scale attention and achieves high detection accuracy.

Compare with Other Detection Frameworks. We study the proposed method compared with other detection frameworks, cooperating with the popular Swin-Transformer (Liu et al., 2021). Table 3 shows the result of state-of-the-art one-stage and two-stage detectors that use a Swin-Tiny backbone. We construct a Deformable D^{2}ETR-SwinT by following the implementation of Swin-Tiny to build our normal stages. Observe that the one-stage and two-stage detectors either consume too much computation or have too many parameters. Our method achieves comparable or even higher performance with fewer computationals and comparable parameters, demonstrating a decent generalization ability.

Table 3: Cooperate with Swin-Tiny. GFLOPs is calculated under 1280x800.

Method	Backbone	Epochs	GFLOPs	Params	AP	AP_{50}	AP_{75}
Cascade Mask RCNN	SwinT	36	745	86	50.5	69.3	54.9
ATSS	SwinT	36	215	36	47.2	66.5	51.3
GFL	SwinT	36	215	36	47.6	66.8	51.7
Sparse RCNN	SwinT	36	172	110	47.9	67.3	52.3
Deformable D^2ETR	SwinT	50	127	46	49.1	67.6	53.3

Table 4: Roadmap of D^2ETR and Deformable D^2ETR.

Method	Epochs	GFLOPs	Params	AP
DETR R50 baseline	50	86	41	34.9
+ decoder-only	50	76 (-10)	33	29.0 (-5.9)
+ PVT2	50	69 (-17)	32	33.2 (-1.7)
+ CECA	50	82 (-4)	35	41.0 (+6.1)
+ IoU-awareness	50	82 (-4)	35	43.2 (+8.3)
+ token labeling	50	82 (-4)	35	43.2 (+8.3)
Deformable DETR R50 baseline	50	173	40	44.5
+ decoder-only	50	88 (-85)	35	39.6 (-4.9)
+ PVT2	50	78 (-95)	30	43.5 (-1.0)
+ CECA	50	88 (-85)	39	45.3 (+0.8)
+ IoU-awareness	50	89 (-84)	39	46.8 (+2.3)
+ bbox refinement and two-stage	50	93 (-80)	40	49.6 (+5.1)
+ token labeling	50	93 (-80)	40	50.0 (+5.5)

5.2 Ablations

We report the ablation study for the design choices in Table 4. The decoder-only architecture brings a notable decrement to the computational complexity. The backbone changing to PVT improves the performance, but cannot completely compensate for the absence of encoder. The proposed CECA scheme increases the detection accuracy at very little cost, especially for the multi-scale detector. With the auxiliary IoU-awareness loss, we improve the performance by enhancing localization quality. Because the object queries in the decoder are randomly initialized and irrelevant to the output of backbone, adding token labeling loss will not improve the performance of D^2ETR. Deformable D^2ETR with two-stage mode activated uses a set of optimal features from backbone to initialize object queries and hence can make use of the token labeling enhanced features. With token labeling, Deformable D^2ETR earns some free accuracy gains.

6 Conclusion

To simplify the pipeline of end-to-end detectors at minimal cost, we present D^2ETR, which is efficient while maintaining high detection accuracy. The core is to build a Transformer that provides both internal and cross-scale feature interaction. The output feature maps from the backbone are fine-fused, thus eliminates the need for independent feature fusing phases. Our method decreases computation consumption significantly and outperforms the original DETR and its variants. We hope that our work will inspire the exploration of the potential of backbone in object detection. In the future, we will investigate the neural architecture search of feature fusing stages.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

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*, pp. 6154–6162, 2018.

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*, pp. 213–229. Springer, 2020.

Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Yuxin Fang, Bencheng Liao, Xinggang Wang, Jiemin Fang, Jiyang Qi, Rui Wu, Jianwei Niu, and Wenyu Liu. You only look at one sequence: Rethinking transformer in vision through object detection. *arXiv preprint arXiv:2106.00666*, 2021.

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.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7036–7045, 2019.

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

Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. *arXiv preprint arXiv:2104.10858*, 2021.

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*, pp. 740–755. Springer, 2014.

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*, pp. 2117–2125, 2017a.

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*, pp. 2980–2988, 2017b.

Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768, 2018.

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.

I Loshchilov and F Hutter. Fixing weight decay regularization in adam, corr, abs/1711.05101. In *Proceedings of the ICLR 2018 Conference Blind Submission, Vancouver, BC, Canada*, volume 30, 2017.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

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.

Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris Kitani. Rethinking transformer-based set prediction for object detection. *arXiv preprint arXiv:2011.10881*, 2020.

Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9627–9636, 2019.

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*, pp. 5998–6008, 2017.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *arXiv preprint arXiv:2106.13797*, 2021.

Shengkai Wu, Xiaoping Li, and Xinggang Wang. Iou-aware single-stage object detector for accurate localization. *Image and Vision Computing*, 97:103911, 2020.

Zhuyu Yao, Jiangbo Ai, Boxun Li, and Chi Zhang. Efficient detr: Improving end-to-end object detector with dense prior. *arXiv preprint arXiv:2104.01318*, 2021.

Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *arXiv preprint arXiv:2011.09315*, 2020.

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.

A Appendix

A.1 Spatial Reduction in Fusing Stage

Given x_i as the feature map of the query scale, $[x_1^*, x_2^*, ..., x_{i-1}^*]$ as the preceding feature maps of the key scales. One layer of i-th fusing stage can be formulated as follows:

$$x_i^* = \text{FFN}(\text{Norm}(A)) + A \tag{10}$$

$$A = \text{SA}\left(x_i, \text{SR}(\{x_k^*\}_{k=1}^{i-1} \cup \{x_i\}), \text{SR}(\{x_k^*\}_{k=1}^{i-1} \cup \{x_i\})\right) + x_i \tag{11}$$

$$\text{SR}(\{x_j\}) = \text{Concat}\left(\{\text{GELU}(\text{Norm}_j(\text{Pool}(x_j)W_j))\}\right) \tag{12}$$

where $\text{SR}(\cdot)$ refers to the spatial reduction operation which utilizes an adaptive poolling layer and enjoys linear computational and memory costs. $W_j \in \mathbb{R}^{C\times C}$ are learnable weights of 1×1 convolutional layer. Norm_j refers to layer normalization.

A.2 Complexity for CECA

In Pyramid Vision Transformer, the linear spatial reduction attention applys an adaptive pool of a fixed size P to the key elements to reduce computational cost. The complexity is $\mathcal{O}(HWP^2C)$. For a multi-scale detector that using S feature maps, suppose the downsampling stride is 2 and h, w are the height and width of the last feature map x_S. The complexity becomes $\mathcal{O}((hw + 4hw + ... + 4^{(S-1)}hw)SP^2C)$ so approximately we have $\mathcal{O}(4^S ShwP^2C)$. As for the CECA, only the latest scale serves as the query elements so the complexity is of $\mathcal{O}(ShwP^2C)$.

A.3 IOU-AWARENESS

IoU-awareness means to align the predicted bounding box with the ground-truth. we use a parameters $\alpha = 0.5$ to control its contribution to the classification score. The final classification score is defined as:

$$Score = CLS^{(1-\alpha)} \times IoU^{\alpha} \tag{13}$$

In the training phase, awareness branches are added to the output of the backbone, the intermediate output and final output of decoder layers. In the inference phase, all the awareness branches are removed except the one on the last decoder layer.

A.4 TOKEN LABELING

With token labeling, each patch token is associated with a location-specific supervision indicating the presence of objects in the corresponding image area. It improves the object localization and recognition capabilities of vision Transformers without computational cost. The token labeling head will be excluded during inference. Figure 4 visualizes the soft token label and predictions from our detector trained with token labeling. Observe that the extracted feature maps are capable of token-level classification.

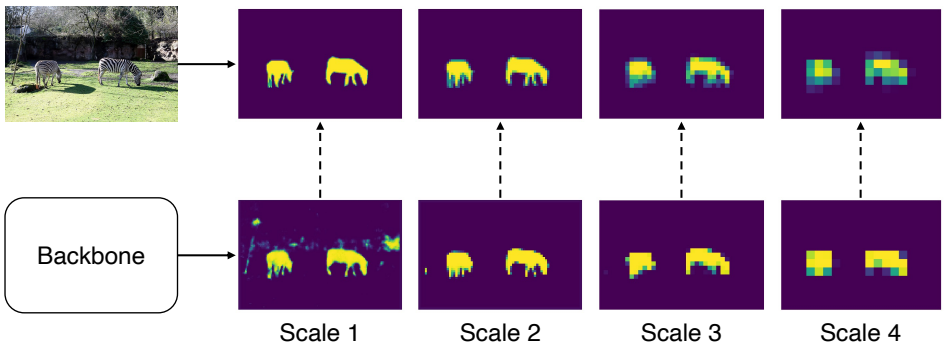


Figure 4: Visualization of token labeling.

A.5 MORE VISUALIZATION ON MULTI-SCALES

To better demonstrate which scale the decoder attends to, we visualize the cross-attention of the decoder on different scales of Deformable DETR and Deformable D^{2}ETR, as shown in Figure 5. The Transformer decoder has 6 decoder layers with one output per layer, we sum them up for readability.

We also plot the average attention on different scales for different object sizes, as shown in Figure 6. We use the first 100 images of the validation set. Following the definition of COCO 2017 benchmark, objects whose pixel area is lower than 32×32, between 32×32 and 96×96, larger than 96×96 are defined as small, medium, and large object, respectively. Notice that the Deformable DETR pays more attention to the low-level scales and cares less about the high-level scales, especially for the small and medium objects. Instead, the proposed Deformable D^{2}ETR can make use of the high-level scale, even when detecting small objects.

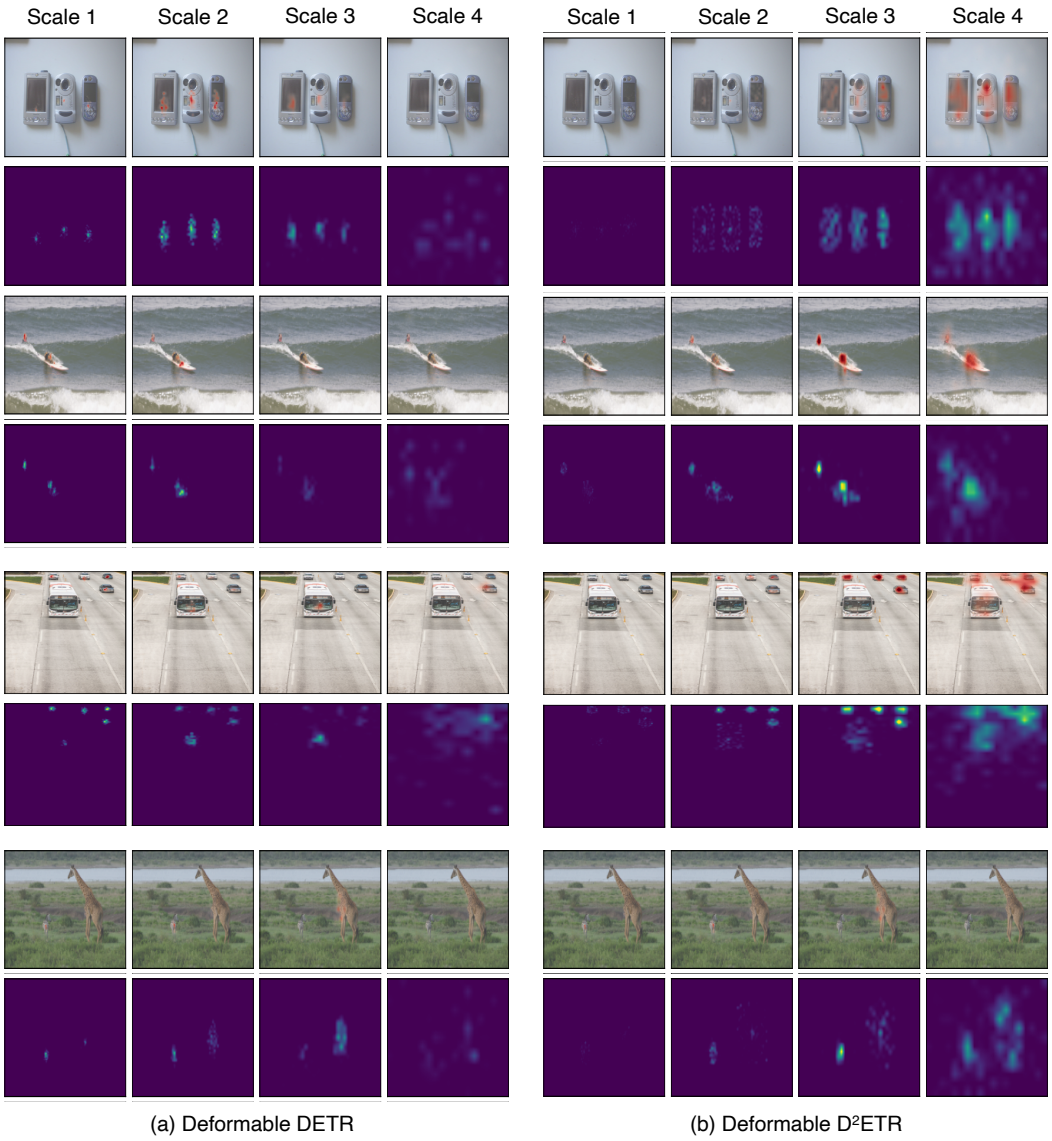


Figure 5: Visualization of cross-attention.

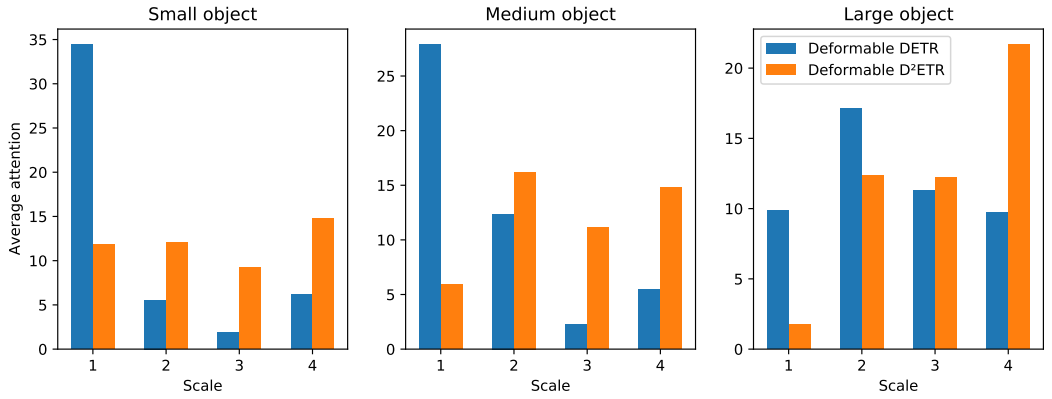


Figure 6: Cross-attention on different scales for different object sizes.

```python
import torch
import torch.nn as nn

class PVTv2_CECA(nn.Module):
    def forward(self, x):
        outs = []
        for i in range(self.num_stages):
            # normal stage
            x = self.patch_embed[i](x)
            for blk in self.block[i]:
                x = blk(x)
            x = self.norm[i](x)
            # fusing stage
            if i >= self.fuse_start_lvl:
                outs.append(self.fuse_proj[len(outs)](x))
                outs_q = outs[-1:]
                outs_k = outs[:-1]
                for blk in self.fuse_block[len(outs)]:
                    outs_q = blk(outs_q, extra_key=outs_k)
                outs_q = self.fuse_norm[len(outs)](outs_q)
                outs = outs_q + outs_k
        return outs

class Block(nn.Module):
    def forward(self, x, extra_key=None):
        x = x + self.drop_path(self.attn(self.norm1(x), x + extra_key))
        x = x + self.drop_path(self.mlp(self.norm2(x)))
        return x

class Attention(nn.Module):
    def forward(self, query, key):
        Q = torch.cat([self.q(x) for x in query])
        KV = [self.kv(self.gelu(self.norm[i](self.conv1x1[i](self.pool(x)
    )))) for i, x in enumerate(key)]
        K = torch.cat([x[0] for x in KV])
        V = torch.cat([x[1] for x in KV])
        attn = ((Q @ K) * self.scale).softmax()
        X = self.proj((attn @ V))
        return X
```

Listing 1: Simplified PyTorch code of CECA based on PVTv2, details such as initialization and reshaping are omitted for readability. The entire code will be made available.

Table 5: Notations in the paper.

Notation	Description
C	Channel number of the input feature map
H	Height of the input feature map
W	Width of the input feature map
x	Input feature map
x_q	Input feature map as query
x_k	Input feature map as key
x_v	Input feature map as value
x_i^j	i-th input feature map after j times of fusions
x_i^*	i-th input feature map after fine-fused
o_q	Object query
σ	Activation function
$\mathbf{H}_i$	Transformer block of i-th stage
W_q	Projection matrix for query
W_k	Projection matrix for key
W_v	Projection matrix for value
P	Adaptive pooling size of spatial reduction
N	Number of object queries
S	Number of feature map fed into the decoder
B	Number of bounding boxes
M	0-1 mask of segmentation annotation
$\hat{y}_i$	Output of the i-th object query
$\hat{b}_i$	i-th predicted bounding box
b_i	i-th target bounding box
$x_i[p, q]$	Position (p, q) on i-th feature map
$t_i[p, q]$	Position (p, q) on i-th soft token label