

# FeatAug-DETR: Enriching One-to-Many Matching for DETRs With Feature Augmentation

Rongyao Fang<sup>1</sup>, Peng Gao<sup>2</sup>, Aojun Zhou<sup>3</sup>, Yingjie Cai<sup>4</sup>, Si Liu<sup>5</sup>, *Senior Member, IEEE*,  
Jifeng Dai<sup>6</sup>, *Member, IEEE*, and Hongsheng Li<sup>7</sup>

**Abstract**—One-to-one matching is a crucial design in DETR-like object detection frameworks. It enables the DETR to perform end-to-end detection. However, it also faces challenges of lacking positive sample supervision and slow convergence speed. Several recent works proposed the one-to-many matching mechanism to accelerate training and boost detection performance. We revisit these methods and model them in a unified format of augmenting the object queries. In this paper, we propose two methods that realize one-to-many matching from a different perspective of augmenting images or image features. The first method is One-to-many Matching via Data Augmentation (denoted as *DataAug-DETR*). It spatially transforms the images and includes multiple augmented versions of each image in the same training batch. Such a simple augmentation strategy already achieves one-to-many matching and surprisingly improves DETR’s performance. The second method is One-to-many matching via Feature Augmentation (denoted as *FeatAug-DETR*). Unlike *DataAug-DETR*, it augments the image features instead of the original images and includes multiple augmented features in the same batch to realize one-to-many matching. *FeatAug-DETR* significantly accelerates DETR training and boosts detection performance while keeping the inference speed unchanged. We conduct extensive experiments to evaluate the effectiveness of the proposed approach on DETR variants, including DAB-DETR, Deformable-DETR, and  $\mathcal{H}$ -Deformable-DETR. Without extra training data, *FeatAug-DETR* shortens the training convergence periods of Deformable-DETR (Zhu et al. 2020) to 24 epochs and achieves 58.3 AP on COCO  $\text{val}2017$  set with Swin-L as the backbone.

**Index Terms**—Detection transformer, one-to-one matching, one-to-many matching, data augmentation, accelerating training.

## I. INTRODUCTION

OBJECT detection is a fundamental task in computer vision, which predicts bounding boxes and categories of objects in an image. In the past several years, deep learning made

Manuscript received 17 February 2023; revised 31 January 2024; accepted 15 March 2024. Date of publication 26 March 2024; date of current version 6 August 2024. This work was supported in part by the National Key R&D Program of China Project under Grant 2022ZD0161100, in part by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd. under the Innovation and Technology Commission (ITC)’s InnoHK, in part by General Research Fund of Hong Kong RGC Project under Grant 14204021. Recommended for acceptance by V. Lempitsky. (*Corresponding author: Hongsheng Li.*)

Rongyao Fang, Aojun Zhou, and Yingjie Cai are with the Chinese University of Hong Kong, Hong Kong SAR, China.

Peng Gao is with Shanghai AI Laboratory, Shanghai 200041, China.

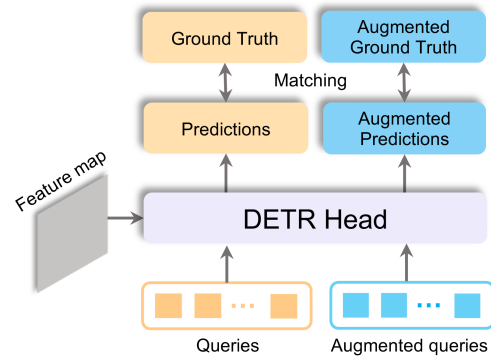
Si Liu is with Beihang University, Beijing 100191, China.

Jifeng Dai is with Tsinghua University, Beijing 100190, China.

Hongsheng Li is with the Centre for Perceptual and Interactive Intelligence, Chinese University of Hong Kong, Hong Kong, China, and also with Shanghai AI Laboratory, Shanghai 200041, China (e-mail: hsl@ee.cuhk.edu.hk).

Digital Object Identifier 10.1109/TPAMI.2024.3381961

(a) Object query augmentation for DETR



(b) Feature map augmentation for DETR (ours)

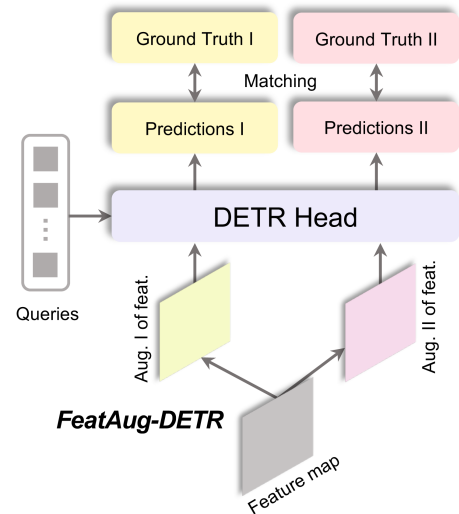


Fig. 1. Previous works achieve one-to-many matching via augmenting object queries [8], [9]. We propose *DataAug-DETR* and *FeatAug-DETR* to achieve one-to-many matching from the perspective of augmenting image features and ground truth objects.

significant success in the object detection task and tens of classic object detectors have been proposed. These classic detectors are mainly based on convolutional neural networks, which include the one-stage detectors [2], [3], [4] and two-stage detectors [5], [6], [7]. One-to-many label assignment is the core design of the classic detectors, where each ground-truth box is assigned to multiple predictions of the detector. With such a one-to-many matching scheme, these frameworks require human-designed non-maximum suppression (NMS) for post-processing and cannot be trained end-to-end.

The DEtection TRansformer (DETR) [10] reveals the potential of using transformer architectures [11] to achieve high performance in object detection. It implements one-to-one matching between the ground truth boxes and the predictions. Various DETR-like frameworks have shown great success. Quite a few follow-up works were proposed to improve DETR by modifying the architectures, such as the architecture of the transformer encoder [1], [12], the decoder [1], [13], [14], [15], and the query designs [16], [17], [18].

Besides improving the DETR architectures, several recent works aim to improve the one-to-one matching mechanism in DETR [10]. The one-to-one matching helps DETR to discard the human-designed NMS for post-processing. In addition, [8] showed that extra one-to-many matching supervisions can lead to faster and better convergence. In the recent Group-DETR [8] and Hybrid Matching [9], extra object queries are used to formulate one-to-many matchings to provide additional supervisions for better training DETRs.

The one-to-many matching mechanism associates each ground truth object with multiple object queries. The object queries interact with the image feature maps containing the objects via cross-attention in the DETR decoder. The one-to-one and one-to-many matchings therefore implicitly conduct assignments between queries and the object features from the spatial feature maps. State-of-the-art Group-DETR and Hybrid Matching enhance one-to-one matching by augmenting extra object queries and inputting them into the matching module. They achieve impressive results in accelerating convergence speed and boosting detection performance.

Our initial observation highlights that a straightforward yet appropriately designed data augmentation scheme (*DataAug-DETR*) can implicitly accomplish one-to-many matching and surprisingly improve DETR performance. By integrating numerous spatially augmented versions of a single image in a single batch, the same objects can be assigned to distinct queries across various augmented images. These one-to-many assignments can greatly enhance detection performance.

Given that DETR queries accumulate information from image feature maps, we propose approximating the impact of spatial image augmentation by applying spatial augmentation to the feature maps, thereby avoiding repeated forwarding of different versions of the same image into the vision backbone. We further propose feature augmentation (*FeatAug-DETR*) for DETR, which spatially shifts and flips feature maps and arranges different versions of a feature map in the same batch, thereby assigning the same object queries to different objects after feature augmentation. This method is a simple yet effective way to enhance DETR performance. The comparison of our feature augmentation and previous object query augmentation is shown in Fig. 1.

We conduct extensive experiments to evaluate the efficiency and effectiveness of *DataAug-DETR* and *FeatAug-DETR*. As a plug-and-play approach, our proposed modules can be easily integrated into different DETR variants. *FeatAug-DETR* significantly accelerates convergence and also improves the performance of various DETR detectors, including

DAB-DETR [18], Deformable-DETR [1], and  $\mathcal{H}$ -Deformable-DETR [9]. *FeatAug-DETR* helps Deformable-DETR [1] with Swin-Large backbone [19] reach 55.8 AP with  $1\times$  training schedule (12 epochs), which is 1.3 AP higher than that without our method (54.5 AP), while keeping the inference FLOPs unchanged. Moreover, *FeatAug-DETR* is compatible with  $\mathcal{H}$ -Deformable-DETR [9] as the feature augmentation realizes one-to-many matchings from a different perspective.

In summary, our contributions are summarized as follows:

- We propose *DataAug-DETR*, which augments each image multiple times and includes the different augmented versions of an image in the same training batch. It boosts DETR’s detection accuracy.
- We further propose feature augmentation (*FeatAug-DETR*) for DETR. It augments the feature maps from the vision backbone and significantly accelerates training compared with *DataAug-DETR*.
- When integrating *FeatAug-DETR* into Hybrid Matching [9], our method achieves 58.3 AP on COCO val2017 with Swin-L backbone and 24 training epochs, surpassing the state-of-the-art performance by 0.5 AP.

## II. RELATED WORK

### A. Classic Object Detectors

Modern object detection models are divided into 2 categories, one-stage detectors, and two-stage detectors. The one-stage detectors [3], [4] predict the positions of the objects relying on the anchors. The two-stage detectors [6] first generate region proposals and then predict the object position w.r.t. the proposals. These methods are both anchor-based methods in which the predefined anchors play an important role in the models. These classic object detectors also need hand-designed operations such as NMS as post-processing, which makes they cannot optimize end-to-end.

### B. Label Assignment in Classic Object Detectors

Assignment between the ground truth objects and training samples is a widely-investigated topic in classic object detectors [2], [3], [6], [7], [20]. Anchor-based detectors [20], [21] utilize Intersection-over-Union (IoU) to apply label assignment. The anchor will be assigned to the maximum IoU ground truth box when the maximum IoU between an anchor and all gt boxes exceeds the predefined IoU threshold. Anchor-free detectors [22], [23] utilize spatial and scale constraints when selecting positive points. The follow-up works [24], [25] propose improvements in a similar direction. The above label assignment methods in classic object detectors are one-to-many matching, which always assigns several object predictions with one ground truth box. Such methods require NMS for post-processing, which makes the detectors hard to train in an end-to-end manner.

### C. Detection Transformer

Carion et al. [10] proposed Detection Transformer (DETR), which introduces the Transformer architecture into the object detection field. They also use bipartite matching to implement

set-based loss and make the framework an end-to-end architecture. These designs remove the handcraft components such as anchors and NMS in the previous classic object detectors. However, DETR still faces the problem of slow convergence speed and relatively low performance. Also, DETR only uses one scale image feature, which lost the benefit of the multi-scale feature which has been proven effective in previous works. The follow-up works proposed improvements to relieve these problems effectively. [26] designed an encoder-only DETR without using a decoder. Anchor-DETR [16] utilizes designed anchor architecture in the decoder to help accelerate training. Conditional-DETR [17] learns conditional spatial query to help cross-attention head to attend to a band containing a distinct region. It significantly shortens the training epochs of DETR from 500 epochs to 50 or 108 epochs. Efficient DETR [27] selects top  $K$  positions from the encoder's prediction to enhance decoder queries. Dynamic Head [28] proposed a dynamic decoder to focus on important regions from multiple feature levels. Deformable-DETR [1], [29], [30] proposed deformable attention and replace the original attention mechanism in DETR. It makes utilizing multi-scale image feature feasible in DETR architecture. The follow-up DAB-DETR [18] uses 4-D box coordinates as queries and updates boxes layer-by-layer in the Transformer decoder part. In the later work, DN-DETR [13] and DINO [31], they use bounding box denoising operation and continue to shorten the training period to  $3\times$  standard training schedule (36 epochs).

#### D. DETR for Pose Estimation

Human pose estimation is a long-standing task in computer vision which requires predicting 2D or 3D locations of anatomical keypoints representing human bodies. Recent top-down approaches led by transformers have shown promising results [32], [33], [34], [35].

Specifically, DETR [10] provides an appealing framework for formulating pose estimation as a set prediction problem. By treating keypoint predictions as objects to be detected, DETR's bipartite matching loss can be directly applied for end-to-end pose learning without the need for additional supervision like association graphs.

Several works have adopted this DETR formulation for multi-person pose estimation. For example, POET [36] introduces a novel loss function and a pose representation to implement pose estimation.

PETR [37] treats pose estimation as hierarchical set prediction using learned pose queries to jointly reason about full-body poses and refine predictions by modeling joint relationships. We select PETR as a representative fully end-to-end framework for multi-person pose and evaluate the effectiveness of our proposed methods on pose estimation DETR.

#### E. DETR for 3D Object Detection

3D object detection aims to identify and localize objects within complex 3D environments represented as point clouds, meshes, or multi-view projections. Transformer-based detectors

like DETR are well-suited for this task due to their strong context modeling capabilities [38], [39], [40], [41], [42].

For multi-view 3D detection, DETR3D [43] modified DETR to operate on multiple perspective images jointly by reasoning about camera projection matrices. Follow-up works like BEVFormer [44] further exploit spatial-temporal context by interacting with bird's eye view representations.

We select a DETR-based detector, 3DETR [41], which set strong performance on popular benchmark ScanNetV2 [45]. Evaluating our method on the framework serves to demonstrate its broad applicability to diverse 3D detection formulations.

#### F. One-to-One Matching

Carion et al. [10] uses bipartite matching to implement one-to-one matching between the object queries and ground truth bounding box. The pipeline computes a pair-wise matching cost depending on the class prediction and the similarity of predicted and ground truth boxes. Then the Hungarian algorithm [46] is applied to find the optimal assignment. The design helps DETR achieve end-to-end training in the object detection field without implementing the hand-designed NMS operations.

However, because of the relatively small number of ground truth boxes in each image (always smaller than 50) compared with the number of predictions which are generally larger than 300. The positive sample supervision provided by one-to-one matching is relatively sparse [8]. It results in a slow convergence speed for the DETR framework. This problem can be effectively revealed by designing one-to-many matching methods.

#### G. One-to-Many Matching

Recently, several works [8], [9] discuss the sparsity positive supervision problem of one-to-one matching and proposed one-to-many matching methods on DETR-liked frameworks. Group-DETR [8] decouples the positives into multiple independent groups and keeps only one positive per gt object in each group. Hybrid Matching [9] combines the original one-to-one matching branch with auxiliary queries that use one-to-many matching loss during training. The principles of these two methods are similar. They provide extra object queries in the decoder to produce more positive supervision in the model. The two methods both effectively accelerate convergence speed and boost performance for DETR.

Our method shares the same principle of generating more positive supervision. But unlike Group-DETR or Hybrid Matching, which implement extra object queries, we augment the images in a batch or the features from the backbone to realize the goal. Our method achieves obvious performance improvement in DETR training. Through experiments, we show that our method continues to obtain further performance boosting when applying previous methods (such as Hybrid Matching) together.

### III. METHOD

#### A. A Brief Revisit of DETR and One-to-Many Matching

In DETR, an input image  $I$  is processed by the backbone  $\mathcal{B}$  to obtain the feature map  $F \in \mathbb{R}^{\hat{C} \times \hat{H} \times \hat{W}}$ . A stack of self-attentions



and feed-forward networks in the DETR encoder transform to obtain the feature maps. The transformer decoder utilizes cross-attention to guide a series of object queries  $Q \in \mathbb{R}^{N \times C}$  to aggregate information from the feature maps and generate object predictions  $P$ , where  $N$  denotes the number of object queries and  $C$  is the query vector dimension. The predictions include the normalized location predictions  $P_l \in \mathbb{R}^{N \times 4}$  and class predictions  $P_c \in \mathbb{R}^{N \times (N_{cls} + 1)}$ , in which  $N_{cls}$  denotes the class number. The one-to-one matching strategy is adopted to associate the predictions  $P$  with ground truth objects  $G$ .

An object query set  $Q = \{q_1, q_2, \dots, q_N\} \in \mathbb{R}^{N \times C}$  is input into the Transformer decoder. The queries aggregate information from image features  $F$  through the cross-attention operations in the Transformer decoder  $\mathcal{D}(\cdot)$  which has  $L$  Transformer layers and outputs object predictions. The predictions of each decoder layer are denoted as  $P^1, P^2, \dots, P^L$  respectively. A bipartite one-to-one matching between the object predictions and the ground truth  $G$  is conducted at each layer. The process is formulated as:

$$P^l = \mathcal{D}^l(Q, F), \quad \mathcal{L}_{\text{one2one}} = \sum_{l=1}^L \mathcal{L}_{\text{Hungarian}}(P^l, G), \quad (1)$$

where  $\mathcal{D}^l(\cdot)$  denotes the  $l$ th layer of the decoder,  $\mathcal{L}_{\text{one2one}}$  denotes the one-to-one matching loss, and  $\mathcal{L}_{\text{Hungarian}}$  denotes the Hungarian matching (bipartite matching) loss at each layer. Since each prediction  $p_i^l \in P^l$  is generated from the object query  $q_i$ , the matchings between ground truths and predictions can be viewed as *the matchings between ground truths  $G$  and object queries  $Q$* .

In order to better supervise DETR and accelerate its training, one-to-many matching schemes were proposed in [8], [9]. Group DETR [8] and Hybrid Matching [9] to augment extra object queries  $\hat{Q}$  and introduce one-to-many matching loss, which achieves significant performance gain. Similar to the formulation of one-to-one matching, the general formulation of one-to-many matching can be defined as:

$$\hat{P}_k^l = \mathcal{D}^l(\hat{Q}_k, F), \quad \mathcal{L}_{\text{one2many}} = \sum_{k=1}^K \sum_{l=1}^L \mathcal{L}_{\text{Hungarian}}(\hat{P}_k^l, \hat{G}_k), \quad (2)$$

where  $K$  denotes the number of Hungarian matching groups. In each matching group, the predictions corresponding to  $k$ -th group of augmented queries  $\hat{Q}_k$  are denoted as  $\hat{P}_k^l$ , and the augmented ground truths are denoted as  $\hat{G}_k$ .

In one-to-many matching, the predictions and ground truths are augmented to generate different groups of positive supervision. Here we discuss the designs of Group-DETR [8] and Hybrid Matching [9] following the above unified formulation (2).

1) *Group-DETR*: Group-DETR utilizes  $K$  separate groups of object queries  $Q_1, \dots, Q_K$  and generates  $K$  groups of predictions for each training image. The same set of ground truth  $G$  applies one-to-one matching to each group of the predictions

respectively. The process can be formulated as:

$$P_k^l = \mathcal{D}^l(Q_k, F), \quad \mathcal{L}_{\text{Group-DETR}} = \sum_{k=1}^K \sum_{l=1}^L \mathcal{L}_{\text{Hungarian}}(P_k^l, G), \quad (3)$$

In Group-DETR, multiple groups of object queries are matched to the same set of ground truths. The augmentation on the object queries helps to boost the model performance.

2) *Hybrid Matching*: In Hybrid Matching, it uses a second group of object queries, which contains  $T$  object queries  $\hat{Q} = \{\hat{q}_1, \hat{q}_2, \dots, \hat{q}_T\}$ . The second group of queries applies one-to-many matching with repeated sets of ground truths  $\hat{G} = \{\hat{G}_1, \hat{G}_2, \dots, \hat{G}_K\}$ , where  $\hat{G}_1 = \hat{G}_2 = \dots = \hat{G}_K = G$ . The process of Hybrid Matching can be formulated as:

$$P^l = \mathcal{D}^l(Q, F), \quad \hat{P}^l = \mathcal{D}^l(\hat{Q}, F), \\ \mathcal{L}_{\mathcal{H}} = \sum_{l=1}^L \mathcal{L}_{\text{Hungarian}}(P^l, G) + \sum_{l=1}^L \mathcal{L}_{\text{Hungarian}}(\hat{P}^l, \hat{G}). \quad (4)$$

Group-DETR and Hybrid Matching both augment the object queries  $\hat{Q}$  to facilitate the training. Considering the unified formulation of one-to-many matching (2), Group-DETR and Hybrid Matching both conduct one-to-many matching via augmenting the ‘‘many’’ query set  $Q$  to match ‘‘one’’ ground truth set but ignore the possibility of jointly augmenting the image features  $F$  and the ground truths  $G$ .

Different from these two methods which augment the object queries  $Q$ , our approach innovatively achieves one-to-many matching focus on augmenting the input images or features while keeping the queries fixed. In our one-to-many matching setting, ‘‘one’’ refers to each object query, and ‘‘many’’ pertains to the different ground truth objects that are matched to the same query across various augmented images or features. This strategy underscores our method’s unique approach in diversifying the query-to-ground truth relationship in object detection frameworks. The pipeline of DataAug-DETR and FeatAug-DETR are shown in Fig. 2.

## B. One-to-Many Matching via Data Augmentation

Our important observation is that one-to-many matching can also be implemented via augmenting the image feature  $F$  and the ground truths  $G$  in (2). We explore conducting spatial augmentation on each image multiple times and include them in the same batch. And we experimentally validate that spatially augmented versions of the same image lead to different query-ground truth assignments.

We conduct a pilot study on COCO train2017 dataset, where we augment every image two times with random flipping and cropping. The random flipping and cropping operations follow the same operations as that in DETR [10] for data augmentation. Such augmented image pairs are then input to a trained Deformable-DETR whose parameters are fixed. The Deformable-DETR has 300 object queries and operates in a one-stage manner. We observe that 95.9% of the corresponding objects in the two augmented images are assigned to different object queries. The remaining 4.1% objects that are assigned

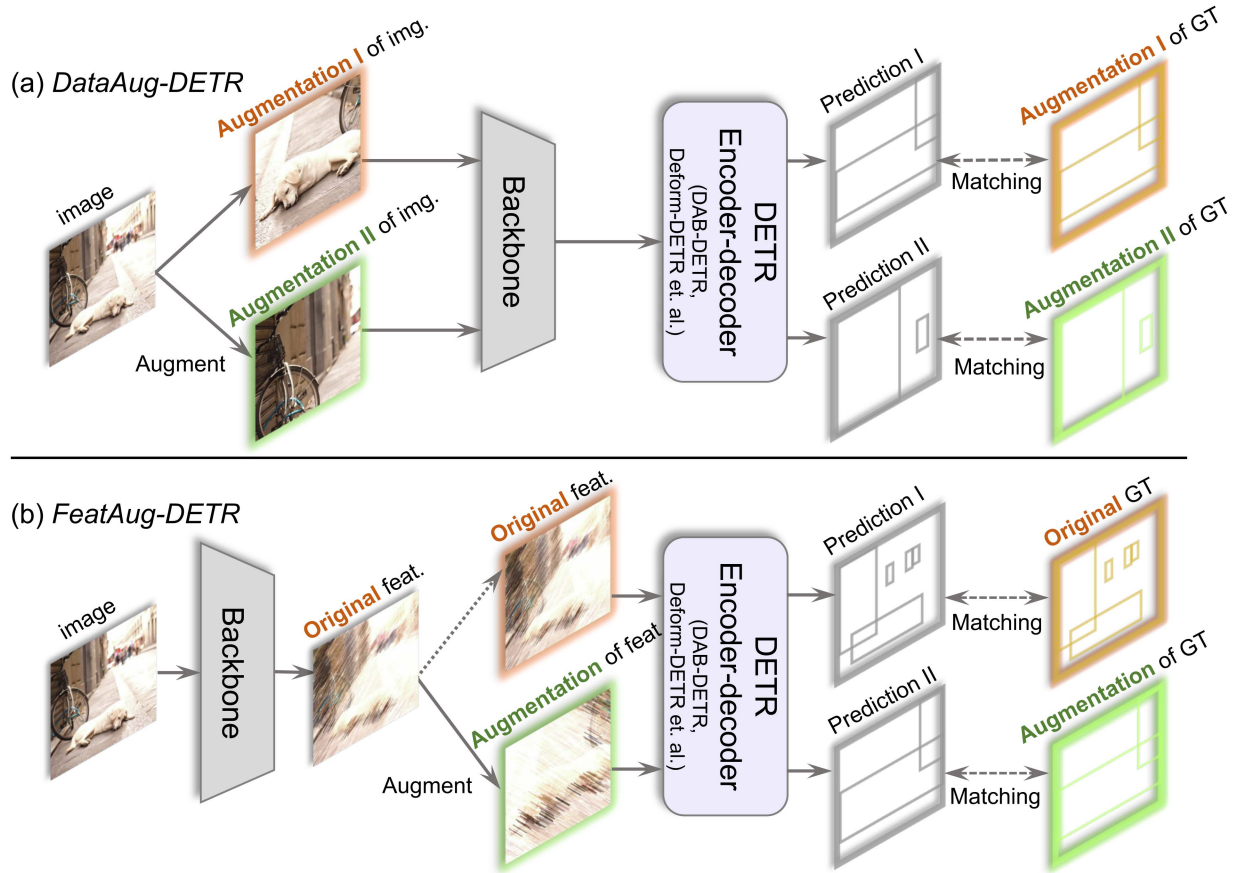


Fig. 2. *DataAug-DETR* augments images several times and include the multiple augmented versions in the same batch. *FeatAug-DETR* augments feature maps from the vision backbone multiple times and include them in the same batch. Our *FeatAug-DETR* can augment both single-scale and multi-scale feature maps. Only single-scale feature maps are shown here for simplicity.

to the same object queries are mostly located at the same relative positions in the two augmented images. More specifically, ground truth bounding boxes with unchanged queries of the two augmented images have a high average IoU of 78.2%. This pilot study shows that, by spatially augmenting each training image in a proper way, the objects can be assigned to different object queries via spatial image augmentation. It is therefore reasonable to take advantage of image spatial transformation to modify  $F$  and  $G$  to achieve one-to-many matching for DETRs.

The above pilot study shows that one-to-many matching can be achieved via spatial data augmentation. We propose *DataAug-DETR*, which conducts spatial image augmentation on each training image and includes them in the same training batch. We adopt the default data augmentation scheme of DETR and Deformable-DETR, which includes a 50% random horizontal flipping and a 50% random cropping, followed by a random resizing to several pre-defined sizes with the short edge ranging from [480, 800]. Assume that each image is spatially augmented for  $N$  times in each training iteration. We denote the data augmentation operation as  $\mathcal{T}_n(\cdot)$ , where  $n$  denotes the  $n$ -th random data augmentation to an image. The  $N$  versions of the image pass through the image feature backbone  $\mathcal{F}$  in DETR and generate  $N$  image features  $\{\hat{F}_n = \mathcal{F}(\mathcal{T}_n(I))\}_{n=1}^N$ . Note that data augmentation is also applied to the ground truth

labels  $G$  and generates  $N$  versions of labels  $\{\hat{G}_n = \mathcal{T}_n(G)\}_{n=1}^N$ . The augmentation process can be formulated as:

$$\hat{F}_n = \mathcal{F}(\mathcal{T}_n(I)), \hat{G}_n = \mathcal{T}_n(G), \text{ for } n = 1, \dots, N. \quad (5)$$

Then by applying the bipartite matching on each augmented image individually, the matching process in (2) becomes:

$$\hat{P}_n^l = \mathcal{D}^l(Q, \hat{F}_n), \mathcal{L}_{\text{DataAug}} = \sum_{n=1}^N \sum_{l=1}^L \mathcal{L}_{\text{Hungarian}}(\hat{P}_n^l, \hat{G}_n), \quad (6)$$

where  $\mathcal{D}^l(\cdot)$  denotes the  $l$ -th Transformer decoder layer in DETR. Note that in the default setting of our *DataAug-DETR*, we use a set of object queries  $Q$  shared with all images. During the Hungarian matching of different augmented versions of an image, the ground truth objects tend to be matched to different object queries.

### C. One-to-Many Matching via Feature Augmentation

In our *DataAug-DETR*, data augmentations are applied to each image multiple times. The augmented  $N$  versions of the same image are encoded by the feature backbone, whose computation cost is considerable as the  $N$  augmented versions

of each image need to be processed by the generally heavy backbone.

Since we choose to perform simple spatial transformations on each training image, the resulting features of the spatially transformed images can also be approximated by conducting the spatial transformations directly on the feature maps  $F$  of each image  $I$ . In this way, each image only goes through the heavy feature backbone once and can still obtain multiple spatially augmented feature maps  $\hat{F}_1, \dots, \hat{F}_N$ . This strategy is much more efficient than *DataAug-DETR* and we name it *FeatAug-DETR*. The process can be formulated as:

$$F = \mathcal{F}(I), \quad \hat{F}_n = \mathcal{E}(\mathcal{T}_n(F)),$$

$$\hat{G}_n = \mathcal{T}_n(G), \quad \text{for } n = 1, 2, \dots, N, \quad (7)$$

where  $\mathcal{E}(\cdot)$  denotes the Transformer encoder of DETR.  $\mathcal{T}_n(\cdot)$  denotes conducting spatial augmentation on the feature map  $F$ . We perform feature augmentation to the output feature of the backbone and before the Transformer encoder  $\mathcal{E}(\cdot)$ , which we experimentally found to achieve better performance. The detailed experiment on the selection of the feature augmentation position can be found in Section IV-L2.

After the augmented feature  $\hat{F}_n$  is obtained, a matching process similar to that of *DataAug-DETR* is applied:

$$\hat{P}_n^l = \mathcal{D}^l(Q, \hat{F}_n), \quad \mathcal{L}_{\text{FeatAug}} = \sum_{n=1}^N \sum_{l=1}^L \mathcal{L}_{\text{Hungarian}}(\hat{P}_n^l, \hat{G}_n), \quad (8)$$

For the specific operation of feature augmentation  $\mathcal{T}_n(\cdot)$ , we investigate feature map flipping or/and cropping.

1) *Feature Map Flipping*: Horizontal flipping is performed on the feature map  $F$  (denoted as *FeatAug-Flip*), which is formulated as:

$$F = \mathcal{F}(I), \quad \hat{F}_1 = \mathcal{E}(F), \quad \hat{F}_2 = \mathcal{E}(\text{Flip}(F)),$$

$$\hat{G}_1 = G, \quad \hat{G}_2 = \text{Flip}(G). \quad (9)$$

After applying *FeatAug-Flip*, the two augmented feature maps  $\hat{F}_1$  and  $\hat{F}_2$  are forwarded to the follow-up modules of DETR and two separate Hungarian matchings are conducted for the two feature maps in the same training batch following (2).

2) *Feature Map Cropping*: Besides the flip operation, random cropping on the feature map is tested (denoted as *FeatAug-Crop*). The process of *FeatAug-Crop* is similar to that of *FeatAug-Flip* but replaces flipping with feature cropping.

The cropping and resizing hyperparameters are the same as the DETR’s original image data augmentation cropping and resizing scheme. Since some state-of-the-art DETR-like frameworks are based on Deformable-DETR, which utilizes multi-scale features from a backbone, our *FeatAug-Crop* augments both single-scale and multi-scale feature maps. Deformable-DETR utilizes multi-scale feature maps of 1/8, 1/16, and 1/32 original resolutions, respectively. In order to crop the features of the three scales, we use the RoIAlign [7] to crop and resize the same region across the three scales. The feature augmentation for single-scale features is the same but only conducts the augmentation on one scale.

When cropping the features with RoIAlign, we find that the features after cropping  $\hat{F}$  become blurry due to the bilinear

interpolation in RoIAlign. It causes a domain gap between the original feature  $F$  and the cropped feature  $\hat{F}$ . When DETR is trained with both types of region features, its detection performance deteriorates. We propose to use extra feature projectors on the cropped features to narrow down the domain gap. Given the three-scale features from the backbone, three individual projectors are adopted to transform the cropped features  $\hat{F}$ , each of which includes a  $1 \times 1$  convolution layer and a group normalization layer [47]. The projectors are only applied on  $\hat{F}$  during training, while the original features  $F$  are still processed by the original detection head. The cropped feature projectors are able to mitigate the domain gap produced by RoIAlign and avoid performance reduction.

3) *Combining Flipping and Cropping*: In *FeatAug-Flip* and *FeatAug-Crop*, the two versions of feature maps of each image, which include the original and the augmented features, are forwarded through the DETR detection head. It is straightforward to combine the above flipping and cropping operations for feature augmentation. We name this combined version *FeatAug-FC*. When applying *FeatAug-FC*, three versions of each image’s feature maps, i.e., the original, flipped, and cropped feature maps, are input into the DETR head.

*DataAug-DETR* and *FeatAug-DETR* are introduced to augment feature maps in the same training batch to realize one-to-many matching for DETRs from a new perspective. Both our methods improve detection performance and *FeatAug-DETR* also significantly accelerates DETR training.

## IV. EXPERIMENT

### A. Dataset and Implementation Details

The experiments are conducted on COCO 2017 object detection dataset [48]. The dataset is split into `train2017` and `val2017`, in which the `train2017` and `val2017` sets contain 118 k and 5 k images, respectively. There are 7 instances per image on average, up to 63 instances in a single image in the training set. We report the standard average precision (AP) results on COCO `val2017`. The DETR frameworks are tested with ResNet-50 [49], Swin-Tiny, and Swin-Large [19] backbones. The ResNet-50 and Swin-Tiny backbones are pre-trained on ImageNet-1K [50] and Swin-Large is pre-trained on ImageNet-22K [50].

We test our proposed methods on top of DAB-DETR [18], Deformable-DETR [1] with tricks (denoted as “Deform-DETR w/ tck.”) from [9], and Hybrid Matching Deformable-DETR ( $\mathcal{H}$ -Deformable-DETR) [9]. The latter two DETR frameworks use additional tricks, including bounding box refinement [1], two-stage [1], mixed query selection [31] and look forward twice [31]. These tricks accelerate the convergence speed and improve the final performance.

We use the L1 loss and GIOU [51] loss for box regression, and focal loss [20] with  $\alpha = 0.25, \gamma = 2$  for classification. As the setting in DETR [10], we apply auxiliary losses after each decoder layer. Similar to Deformable-DETR [1], we add extra intermediate losses after the query selection module, with the same components as for each decoder layer. We adopt the loss

TABLE I  
MAIN RESULTS OF PROPOSED DATA AUGMENTATION AND FEATURE AUGMENTATION ON VARIOUS DETR FRAMEWORKS

Method	Backbone	Tricks	#queries	#epochs	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
DETR-DC5 [10]	R50	✗	300	500	43.3	63.1	45.9	22.5	47.3	61.1
Conditional-DETR-DC5 [17]	R50	✗	300	108	45.1	65.4	48.5	25.3	49.0	62.2
DAB-DETR-DC5 [18]	R50	✗	300	50	45.7	66.2	49.0	26.1	49.4	63.1
+FeatAug-FC	R50	✗	300	36	46.1 (↑0.4)	66.6	49.5	27.7	50.0	62.8
+FeatAug-FC	R50	✗	300	50	47.1 (↑1.4)	67.4	50.8	27.9	50.9	64.3
Deformable-DETR-One-Stage [1]	R50	✗	300	50	44.5	63.6	48.7	27.1	47.6	59.6
Deformable-DETR [1]	R50	✗	300	50	46.9	65.6	51.0	29.6	50.1	61.6
DAB-Deformable-DETR [18]	R50	✗	300	50	46.9	66.0	50.4	29.1	49.8	62.3
DN-Deformable-DETR [13]	R50	✗	300	12	43.4	61.9	47.2	24.8	46.8	59.4
DN-Deformable-DETR [13]	R50	✗	300	50	48.6	67.4	52.7	31.0	52.0	63.7
DINO-Deformable-DETR [31] <sup>†</sup>	R50	✓	900	12	47.9	65.3	52.1	31.2	50.9	61.9
DINO-Deformable-DETR [31] <sup>†</sup>	R50	✓	900	36	50.5	68.3	55.1	32.7	53.9	64.9
DINO-Deformable-DETR [31] <sup>†</sup>	Swin-L	✓	900	12	56.8	75.6	62.1	39.9	60.4	73.3
DINO-Deformable-DETR [31] <sup>†</sup>	Swin-L	✓	900	36	57.8	76.5	63.2	40.6	61.8	73.5
Deformable-DETR w/ tck. [1]	R50	✓	300	12	47.0	65.3	50.9	30.7	50.0	60.9
+FeatAug-FC	R50	✓	300	12	48.7 (↑1.7)	67.1	53.3	31.2	52.1	63.2
Deformable-DETR w/ tck. [1]	R50	✓	300	36	49.0	67.6	53.3	32.7	51.5	63.7
+DataAug	R50	✓	300	48	50.0 (↑1.0)	68.6	54.6	33.3	52.9	64.5
+FeatAug-FC	R50	✓	300	24	49.9 (↑0.9)	68.3	54.7	32.5	52.9	65.1
Deformable-DETR w/ tck. [1]	Swin-T	✓	300	12	49.3	67.8	53.4	31.6	52.4	64.4
+FeatAug-FC	Swin-T	✓	300	12	50.9 (↑1.6)	69.4	55.6	33.1	54.3	66.1
Deformable-DETR w/ tck. [1]	Swin-T	✓	300	36	51.8	70.9	56.5	34.6	55.1	67.9
+DataAug	Swin-T	✓	300	48	53.3 (↑1.5)	72.1	58.5	35.9	56.7	68.4
+FeatAug-FC	Swin-T	✓	300	24	52.7 (↑0.9)	71.3	57.5	35.1	56.6	67.8
Deformable-DETR w/ tck. [1]	Swin-L	✓	300	12	54.5	74.0	59.2	37.0	58.6	71.0
+FeatAug-FC	Swin-L	✓	300	12	55.8 (↑1.3)	75.5	61.1	39.2	60.2	72.1
Deformable-DETR w/ tck. [1]	Swin-L	✓	300	36	56.3	75.7	61.5	39.1	60.3	71.9
+DataAug	Swin-L	✓	300	48	57.0 (↑0.7)	76.3	62.2	40.4	60.9	73.3
+FeatAug-FC	Swin-L	✓	300	24	57.1 (↑0.8)	76.4	62.5	41.1	60.9	73.3
+FeatAug-FC <sup>†</sup>	Swin-L	✓	900	24	57.6	76.7	63.1	41.1	61.5	73.8
H-Deformable-DETR [9]	R50	✓	300	12	48.7	66.7	53.4	31.4	51.8	63.6
+FeatAug-Flip	R50	✓	300	12	49.4 (↑0.7)	67.5	53.7	31.9	52.7	64.3
H-Deformable-DETR [9]	R50	✓	300	36	50.0	68.1	54.6	32.5	53.1	65.0
+FeatAug-Flip	R50	✓	300	24	50.4 (↑0.4)	68.7	54.9	32.7	53.6	65.2
H-Deformable-DETR [9]	Swin-L	✓	300	12	55.9	75.1	61.0	39.3	59.9	72.1
+FeatAug-Flip	Swin-L	✓	300	12	56.4 (↑0.5)	75.6	61.8	40.0	60.4	72.4
H-Deformable-DETR [9]	Swin-L	✓	300	36	57.1	76.3	62.7	40.2	61.6	73.3
+FeatAug-Flip	Swin-L	✓	300	24	57.6 (↑0.5)	76.6	63.1	40.4	61.9	73.9
H-Deformable-DETR [9] <sup>†</sup>	Swin-L	✓	900	36	57.9	76.9	63.8	42.5	62.0	73.5
+FeatAug-Flip <sup>†</sup>	Swin-L	✓	900	24	<b>58.3</b> (↑0.4)	77.1	64.0	41.7	62.4	73.9

Tricks: tricks described in Section IV-A. <sup>†</sup>: keep 300 instead of 100 predictions for evaluation.

coefficients: 2.0 for classification loss, 5.0 for L1 loss, and 2.0 for GIOU loss, which is the same as [9].

Each tested DETR framework is composed of a feature backbone, a Transformer encoder, a Transformer decoder, and two prediction heads for boxes and labels. All Transformer weights are initialized with Xavier initialization [52]. In the experiments, we use 6 layers for both the Transformer encoder and decoder. The hidden dimension of the Transformer layers is 256. The intermediate size of the feed-forward layers in the Transformer blocks is 2048, which follows the settings of [9]. The MLP networks for box and label predictions share the same parameters across different decoder layers. We use 300 object queries in the decoder. We use AdamW [53] optimizer with a weight decay  $10^{-4}$ . We use an initial learning rate of  $2 \times 10^{-4}$  for the Deformable DETR head and a learning rate of  $2 \times 10^{-5}$  for the backbone, which is the same as those in [1]. A 1/10 learning rate

drop is applied at the 11th, 20th, and 30th epochs for the 12, 24, and 36 epoch settings, respectively. The model is trained without dropout. The training batch size is 16 and the experiments are run on 16 NVIDIA V100 GPUs. During validation, we select 100 predicted boxes and labels with the largest classification logits for evaluation by default.

## B. Main Results

Our *DataAug-DETR* and *FeatAug-DETR* methods are compatible with most DETR-like frameworks. The results on COCO val2017 set are shown in Table I. The DAB-DETR-DC5-*FeatAug-FC* denotes applying *FeatAug-FC* on top of DAB-DETR [18] with R50 dilated C5-stage image features [54], Deformable-DETR w/ tck. -*FeatAug-FC* denotes *FeatAug-FC*



TABLE II  
FEATAUG-FC VERSUS FEATAUG-FLIP ON TOP OF HYBRID MATCHING  
DEFORMABLE-DETR

Method	Backbone	#epochs	AP	AP <sub>50</sub>	AP <sub>75</sub>
$\mathcal{H}$ -Deformable-DETR [9]	R50	24	50.0	68.1	54.6
+FeatAug-FC	R50	24	50.1	68.2	54.6
+FeatAug-Flip	R50	24	50.4	68.7	54.9

on top of Deformable-DETR with tricks, and  $\mathcal{H}$ -Deformable-DETR-FeatAug-Flip denotes FeatAug-Flip on top of Hybrid matching Deformable-DETR. We experimentally found that, when integrating with Hybrid Matching, FeatAug-Flip achieves better performance than FeatAug-FC as shown in Table II. Thus, we report the results of Hybrid Matching with FeatAug-Flip and others with FeatAug-FC.

In the table, we also report the performances of some previous representative DETR variants. The compared methods include single-scale and multi-scale detectors. The Deformable-DETR ones, which utilize multi-scale features, generally achieve better performances than single-scale detectors.

We compare the performance gain of our DataAug-DETR and FeatAug-DETR on top of the baselines. DataAug-DETR is evaluated on top of Deform-DETR w/ tck. [1]. FeatAug-DETR is evaluated on DAB-DETR [18], Deform-DETR w/ tck., and  $\mathcal{H}$ -Deformable-DETR [9]. DataAug-DETR (48 epochs training) improves the performance by about 1.0 AP compared with the Deform-DETR w/ tck. (36 epochs training) baseline, while the performance of the baseline trained with 48 epochs degrades. In each epoch of DataAug-DETR, we only train  $1/N$  of the whole training set, where  $N$  is the augmentation times of DataAug-DETR. Thus, the training time per epoch of DataAug-DETR is the same as that of the baseline. The detailed investigation of the convergence speed of DataAug-DETR is shown in Section IV-H.

On the single-scale DAB-DETR with ResNet-50 backbone, FeatAug-FC improves the performance for 1.4 AP with 50 training epochs and reaches 47.1 AP, which makes the single-scale detector’s performance better than the multi-scale Deformable-DETR [1] (46.9 AP). It is shown that the AP<sub>M</sub> and AP<sub>L</sub> of DAB-DETR-DC5-FeatAug-FC exceed those of Deformable-DETR by large margins (0.8 AP and 2.7 AP, respectively). When shortening the training epochs to 36 epochs, FeatAug-FC still achieves the performance of 46.1 AP, which is 0.4 AP higher than the DAB-DETR-DC5 baseline trained with 50 epochs.

On top of the multi-scale Deform-DETR w/ tck. baseline, FeatAug-FC not only achieves around 1.0 AP gain after convergence but also shortens the training epochs (24 epochs versus 36 epochs) on all the three tested backbones. Our FeatAug-FC integrated into Deform-DETR w/ tck. on Swin-Large backbone achieves 57.6 AP, which is comparable with the state-of-the-art DINO-Deformable-DETR framework.

The work Hybrid Matching (denoted as  $\mathcal{H}$ ) [9] also tackles the one-to-many matching problem and achieves state-of-the-art performance. We also test our FeatAug-DETR on top of Hybrid Matching with Swin-Large backbone and achieve the performance of 58.3 AP, which is 0.5 AP higher than the previous state-of-the-art DINO. In experiments, we adopt the default hyperparameter settings of Hybrid Matching as [9].

TABLE III  
EVALUATION ON PETR FOR POSE ESTIMATION

Method	Backbone	#epochs	AP	AP <sub>50</sub>	AP <sub>75</sub>
PETR [37]	ResNet50 [49]	100	68.8	87.5	76.3
+FeatAug-Flip	ResNet50 [49]	100	70.5	88.6	77.5
+FeatAug-FC	ResNet50 [49]	100	70.8	88.7	78.0
PETR [37]	Swin-Large [19]	100	73.1	90.7	80.9
+FeatAug-Flip	Swin-Large [19]	100	74.5	91.1	81.7
+FeatAug-FC	Swin-Large [19]	100	74.8	91.3	82.5

TABLE IV  
EVALUATION ON 3DETR FOR 3D DETECTION

Method	#epochs	AP <sub>25</sub>	AP <sub>50</sub>
3DETR-m [41]	180	58.70	33.60
+FeatAug-Flip	180	59.61	33.83
3DETR-m [41]	720	63.70	44.58
+FeatAug-Flip	720	64.69	46.52

### C. Multi-Person Pose Estimation Results

To evaluate the generalization of our methods to multi-person pose estimation, we apply FeatAug-Flip and FeatAug-FC to PETR (Pose Estimation with TRansformers) [37] on the COCO pose estimation dataset. As shown in Table III, using ResNet50 [49] and Swin-Large [19] backbones, FeatAug-Flip improves over the PETR baseline by 1.7 AP and 1.4 AP respectively after 100 training epochs. With FeatAug-FC, gains of 2.0 AP and 1.7 AP are achieved for ResNet50 and Swin-Large respectively. These consistent improvements verify that our proposed techniques can effectively extend to pose estimation DETR models.

### D. 3D Object Detection Results

We evaluate FeatAug-Flip on 3DETR [41] for 3D detection on ScanNetV2 [45]. We flip the scenes towards the  $x$ -axis or  $y$ -axis as the feature augmentation. As shown in Table IV, the baseline achieves 58.70 AP<sub>25</sub> at 180 epochs and 63.70 AP<sub>25</sub> at 720 epochs. With FeatAug-Flip, gains of 0.9 AP<sub>25</sub> and 1.0 AP<sub>25</sub> are achieved at 180 and 720 epochs respectively. This demonstrates that our approach can generalize to enhance DETR-based 3D object detection as well.

### E. Quantitative Analysis of One-to-Many Matching

To provide direct evidence that our proposed augmentations achieve one-to-many matching, we compute the ratio of object queries that are assigned to different ground truth boxes after applying DataAug-DETR and FeatAug-DETR and denote it as match difference ratio. Different from prior works that assign multiple queries to each ground truth, our approach keeps the queries fixed and matches them to different ground truths across augmented inputs.

As shown in Table V, with DAB-DETR-DC5 [18], DataAug-DETR and FeatAug-Flip achieved match difference ratios of 93.6% and 91.0%, respectively. Similarly, in the one-stage Deformable-DETR [1] framework, these ratios were 95.9% for DataAug-DETR and 92.1% for FeatAug-Flip. The match



TABLE V  
OBJECT QUERY MATCHING DIFFERENCE RATIO IN TWO AUGMENTATION

Framework	Method	Backbone	Match Diff. Ratio (%)
DAB-DETR-DC5 [18]	<i>DataAug-DETR</i>	R50	93.6
DAB-DETR-DC5 [18]	<i>FeatAug-Flip</i>	R50	91.0
Deform-one-stage [1]	<i>DataAug-DETR</i>	R50	95.9
Deform-one-stage [1]	<i>FeatAug-Flip</i>	R50	92.1

TABLE VI  
PERFORMANCE OF *DATAAUG-DETR* WITH NON-SPATIAL TRANSFORMATION

Method	Backbone	#epochs		
		12	24	36
Deform-DETR w/ tck.	R50	47.0	48.7	49.0
+ <i>DataAug</i>	R50	46.7	49.1	49.8
+ <i>DataAug-Resizing</i>	R50	45.2	48.1	48.9

difference ratio reflects the proportion of object queries paired with different ground truth boxes post-augmentation. These high ratios are instrumental in demonstrating that a substantial portion of queries are consistently assigned to unique ground truths in augmented data, thus providing a quantitative measure of our one-to-many matching efficacy.

Besides discussing the match difference ratios achieved by our methods, it is instructive to consider the metric in the context of baseline and prior one-to-many methods. Baseline models, which do not incorporate augmentations, inherently exhibit a 0% match difference ratio. On the other hand, methods like Group-DETR and Hybrid Matching, which apply new arbitrary queries for one-to-many matching, can be regarded as achieving 100% match difference ratios. However, since our approach is complementary to these methods, the difference ratios are not directly comparable. Our methods offer a novel perspective on achieving effective one-to-many matching.

#### F. Non-Spatial versus Spatial Transformation for DataAug-DETR

As discussed in Section III-B, the object queries' assignments with the ground truths are sensitive to position changes. In other words, when an augmentation changes the relative position of objects, these objects almost always match different queries compared to the original image/feature. Our proposed flipping and cropping augmentations are both spatial transformations that change relative object positions. There are also other widely used data augmentation methods that do not change the objects' relative positions, such as image random resizing. In this section, we also test applying only image random resizing in our *DataAug-DETR* method, which applies random resizing of each image several times in the same batch. We compare the non-spatial transformation with the Deform-DETR w/ tck. baseline and our *DataAug-DETR* with default settings. In the experiments, the augmentation times for each image  $N = 2$ .

As shown in Table VI, the performance of only applying image resizing in *DataAug-DETR* is even worse than the baseline and our proposed default *DataAug-DETR* setting. The model converges slower and leads to worse performance than the

TABLE VII  
CONVERGENCE ANALYSIS OF *DATAAUG-DETR*

Method	Backbone	#epochs			
		12	24	36	48
Deform-DETR w/ tck.	R50	47.0	48.7	<b>49.0</b>	48.3
+ <i>DataAug-N = 2</i>	R50	46.7	49.1	49.8	<b>50.0</b>
+ <i>DataAug-N = 3</i>	R50	45.6	48.4	49.4	<b>49.9</b>

baseline. This experiment shows that spatial transformations such as flipping and cropping are crucial in the effectiveness of *DataAug-DETR*. It also shows the rationality of the flipping and cropping feature augmentation in *FeatAug-DETR*.

#### G. Comparison With Normal Data Augmentation and Extra Epochs

To clarify that the performance gains of *DataAug-DETR* and *FeatAug-DETR* are not simply due to extra data augmentation, we compare to a baseline that also uses the flipping and cropping augmentation schedule but with additional training epochs. As shown in Fig. 3 (left) and Table VII, the performance of the baseline actually decreases when trained for more epochs (48 epochs). In contrast, with our *DataAug-DETR* and *FeatAug-DETR* methods, performance continues improving with more training. This demonstrates that the additional gains are not merely from more augmentation, but rather from the one-to-many matching brought by the proposed augmentation strategies.

#### H. Convergence Analysis of DataAug-DETR

In the following analysis experiments, unless otherwise specified, we test our proposed method and evaluate its different designs on top of the Deform-DETR w/ tck. and ResNet-50 backbone and treat it as our experiment baseline.

When keeping the same 16 batch size as the baseline, *DataAug-DETR* can be viewed as changing the order of training data compared with the ordinary training pipeline. Here we investigate the convergence speed of *DataAug-DETR* AP-epoch curves.

As shown in Fig. 3 (left) and Table VII, applying *DataAug-DETR* improves the final converged performance. The baseline converges with 36 epochs and achieves 49.0 AP. Further training hurts the model as the performance drops when trained for 48 epochs. After applying *DataAug-DETR* with augmentation times  $N = 2$ , its performance at 36 epochs is 0.8 AP higher than the baseline. The performance continues to improve with a 48-epoch training scheme. It is also observed that *DataAug-DETR* slightly slows down convergence in early epochs.

This slower early convergence can be attributed to the fact that *DataAug-DETR* sees fewer unique images per epoch compared to the baseline. In *DataAug-DETR*, each image is augmented multiple times before being input to the model, meaning fewer unique original images are processed per epoch. Therefore, *DataAug-DETR* takes longer to accumulate gradients over the full diversity of the dataset initially. This analysis explains the

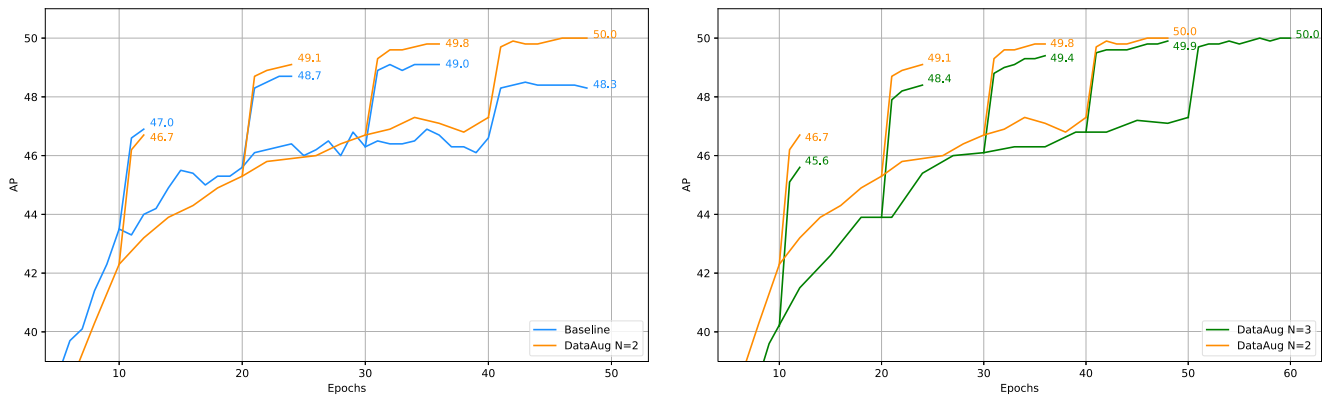


Fig. 3. Left: Training AP curves of Deform-DETR w/ tck. baseline and *DataAug-DETR* with  $N = 2$ . Right: Training AP curves of *DataAug-DETR* with  $N = 2$  and  $N = 3$ . It shows that augmenting each image for more than two times results in slower convergence and is non-beneficial to the final performance.

TABLE VIII

RESULTS OF DIFFERENT *FEATAUG-DETR* METHODS ON DEFORM-DETR w/ TCK

Method	#ep.	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
Deform-DETR w/ tck.	12	47.0	65.3	50.9	30.7	50.0	60.9
Deform-DETR w/ tck.	36	49.0	67.6	53.3	32.7	51.5	63.7
+ <i>FeatAug-Flip</i>	12	48.4	66.7	52.8	31.7	51.4	64.0
+ <i>FeatAug-Flip</i>	24	49.6	68.1	54.1	31.6	53.1	64.8
+ <i>FeatAug-Crop</i>	12	48.1	66.3	52.3	30.7	51.3	63.2
+ <i>FeatAug-Crop</i>	24	49.6	68.1	54.1	33.2	52.6	64.7
+ <i>FeatAug-FC</i>	12	48.7	67.1	53.3	31.2	52.1	63.2
+ <i>FeatAug-FC</i>	24	<b>49.9</b>	68.3	54.7	32.5	52.9	65.1

TABLE IX

COMPUTATIONAL CONSUMPTION OF *FEATAUG-DETR* ON DEFORM-DETR w/TCK. WITH 16 NVIDIA V100 GPUS

Method	Backbone	Time (per epoch)	Time (total)
Deform-DETR w/ tck.	R50	40min	24h (36epochs)
+ <i>FeatAug-Flip</i>	R50	60min	24h (24epochs)
Deform-DETR w/ tck.	Swin-T	50min	30h (36epochs)
+ <i>FeatAug-Flip</i>	Swin-T	70min	28h (24epochs)
Deform-DETR w/ tck.	Swin-L	140min	84h (36epochs)
+ <i>FeatAug-Flip</i>	Swin-L	160min	64h (24epochs)
+ <i>FeatAug-FC</i>	Swin-L	180min	72h (24epochs)

slower early convergence observed for *DataAug-DETR* compared to the baseline.

We also investigate the augmentation number  $N$  with *DataAug-DETR*, Fig. 3 (right) shows that the convergence is slower with a larger augmentation number, while the final performance becomes saturated after  $N \geq 2$ . Thus we adopt  $N = 2$  as the default setting unless otherwise specified.

### I. Comparison of Different Feature Augmentation Operators

We compare the performance of the proposed *FeatAug-Flip*, *FeatAug-Crop*, and *FeatAug-FC* on the Deform-DETR w/ tck. baseline. The results are listed in Table VIII.

As shown in the table, all of our *FeatAug-DETR*'s results are better than that of the baseline. *FeatAug-Flip* is 0.3 AP higher than *FeatAug-Crop* with 12 training epochs. The stronger *FeatAug-FC* is 0.3 AP better than *FeatAug-Flip* and *FeatAug-Crop* with 24 epochs and reaches 49.9 AP.

The results show that *FeatAug-Flip* converges faster than *FeatAug-Crop*, while their convergence performance is similar. *FeatAug-FC* provides further performance improvement compared with *FeatAug-Flip* and *FeatAug-Crop*, while it also requires extra training time for epoch. Thus, *FeatAug-Flip* is suitable for models with relatively small backbones (e.g., R50 and Swin-Tiny). *FeatAug-FC* is preferred when training with large backbones (e.g., Swin-Large).

### J. Convergence Analysis of FeatAug-DETR

In *FeatAug-DETR*, the Transformer encoder and decoder process several augmented features during training, which produce extra computational consumption. However, such an increase in computation is invariant to the scale of the backbone. When training with large-scale backbones, the extra computational consumption is relatively small compared with the backbone. The training time per epoch comparisons with different backbones is shown in Table IX.

The table reports the training time of the Deform-DETR w/ tck. baseline and that integrated with *FeatAug-Flip*. The training time of *FeatAug-Crop* is similar to *FeatAug-Flip*. Our training process uses 16 NVIDIA V100 GPUs with a batch size of 16. As shown in Table IX, the increase in the computation time is invariant with the size of the backbone. The proportion of the increased computation time is only about 15% when using the Swin-L backbone.

The training AP curves of *FeatAug-DETR* are shown in Fig. 4. The left figure shows the curves w.r.t. training epochs. Our *FeatAug-DETR* consistently surpasses the Deform-DETR w/ tck. baseline by a large margin and shortens the training epochs from 36 epochs to 24 epochs. Considering our *FeatAug-DETR* requires extra computation consumption in each epoch, we also show the training AP curves w.r.t. GPU  $\times$  Hours in the right figure. It shows *FeatAug-DETR* still achieves better performance than the baseline in terms of detection accuracy and convergence speed. Especially when the backbone scale is larger, the performance gain is more obvious.

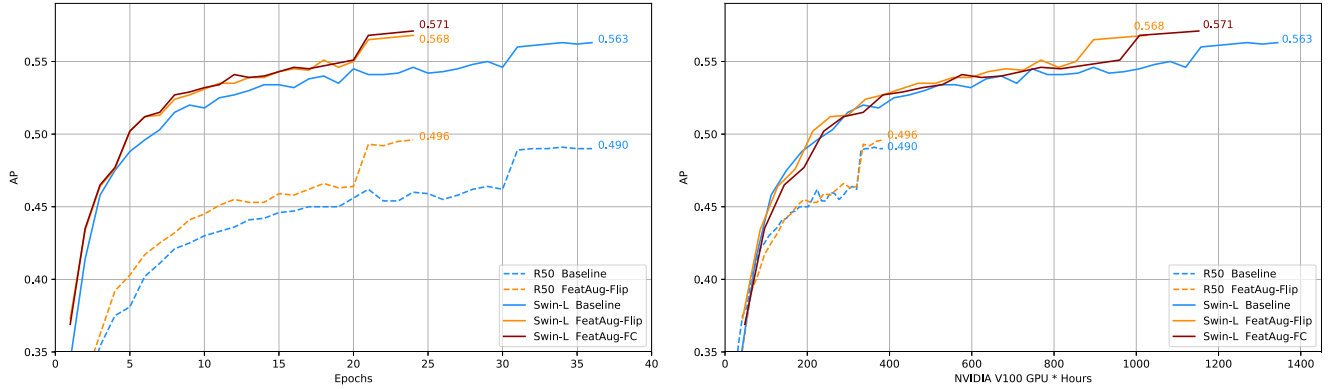


Fig. 4. Left: Training AP curves w.r.t training epochs of Deform-DETR w/ tck. and that integrated with our *FeatAug-DETR*. Right: Training AP curves w.r.t GPU  $\times$  Hours of Deform-DETR w/ tck. and that integrated with our *FeatAug-DETR*.

TABLE X  
RESULTS WITH SHORT TRAINING EPOCHS

Method	Backbone	#epochs	AP	AP <sub>50</sub>	AP <sub>75</sub>
Deformable-DETR w/ tck. [1]	R50	12	47.0	65.3	50.9
+ <i>DataAug</i>	R50	12	46.7	64.9	50.8
+ <i>FeatAug-Flip</i>	R50	12	48.4	66.8	52.5
+ <i>FeatAug-FC</i>	R50	12	48.7	67.1	53.3
$\mathcal{H}$ -Deformable-DETR [9]	R50	12	48.7	66.7	53.4
+ <i>DataAug</i>	R50	12	48.6	66.9	53.1
+ <i>FeatAug-Flip</i>	R50	12	49.4	67.5	53.7

### K. Performances With Short Training Epochs

In Table X, we provide experiments on the Deformable-DETR w/ tck. In [1] and  $\mathcal{H}$ -Deformable-DETR [9] variants trained for 12 epochs to validate the effectiveness of our methods in short-epoch training scenarios.

It can be observed that both our proposed *FeatAug-Flip* and *FeatAug-FC* methods significantly improve performance over the base models with only 12 epochs of training. For example, *FeatAug-FC* boosts Deformable-DETR w/ tck. by 1.7 AP. This demonstrates the capability of our methods to deliver gains in the limited data regime.

While *DataAug-DETR* does lag slightly behind the base model in the 12 epoch setting, we emphasize that it is designed for longer epoch training where it can accumulate gradients over augmented data more effectively. As discussed in Section IV-H, *DataAug-DETR* sees fewer unique images per epoch, so short epochs do not play to its strengths. However, in Section IV-B, we show *DataAug-DETR* can boost performance given adequate training epochs.

In summary, our proposed techniques, especially *FeatAug-Flip/FC*, are broadly effective at improving detection performance with limited training data. The experiments on 12 epoch training further verify the efficiency of our methods.

### L. Ablation Studies

1) *Evaluation on One-Stage Deformable-DETR*: Since we mainly evaluate our method based on Deform-DETR w/ tck., which consists of tricks that boost performance. In order to show

TABLE XI  
EVALUATION ON ONE-STAGE DEFORMABLE-DETR

Method	#ep.	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
One-stage	50	44.5	63.6	48.7	27.1	47.6	59.6
+ <i>FeatAug-FC</i>	36	45.9	64.8	50.3	27.7	49.3	60.6
+ <i>FeatAug-FC</i>	50	<b>46.3</b>	65.3	50.6	28.0	49.7	61.1

TABLE XII  
ABLATION ON INPUT FEATURE SELECTION IN *FEATAUG-DETR*

Method	#ep.	AP	AP <sub>50</sub>	AP <sub>75</sub>
Deform-DETR w/ tck.	12	47.0	65.3	50.9
+ <i>FeatAug-Flip-Encoder</i>	12	46.5	64.8	50.6
+ <i>FeatAug-Flip-Default</i>	12	<b>48.4</b>	66.7	52.8

that our methods are independent of these tricks and can still improve various DETR variants' performance, we also test our method on top of the one-stage Deformable-DETR, which is its original version. The results are shown in Table XI.

Our method accelerates its convergence speed. *FeatAug-FC* trained for 36 epochs is 1.4 AP better than the one-stage Deformable-DETR trained for 50 epochs. With the same 50 training epochs, *FeatAug-FC* reaches 46.3 AP, which is 1.8 AP better than the baseline.

2) *Input Feature of FeatAug-DETR*: In our *FeatAug-DETR*, we augment the image feature from the backbone and input the augmented features into the Transformer encoder. However, in the Deform-DETR w/ tck. baseline, the features after the Transformer encoder can also be chosen as the alternative for augmentation. To analyze which feature is better as the input of *FeatAug-DETR*, we test them and the results are shown in Table XII. *FeatAug-DETR* on the feature after the Transformer encoder is denoted as *FeatAug-Encoder* and the augmentation on the feature directly from the backbone is denoted as *FeatAug-Default*.

As shown in the table, the feature augmentation on the encoder feature actually hurts the performance and is even worse than that of the baseline. Since the Transformer encoder enhances the image features with positional information. If the feature augmentation is applied after the Transformer encoder, it would



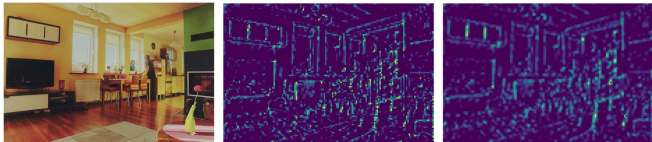


Fig. 5. Visualization of randomly selected one channel feature of the output of vision backbone. It visualizes the feature domain gap caused by RoIAlign, we adopt projectors after the cropped features to mitigate the domain gap. Left: Original input image. Middle: Original feature from vision backbone. Right: Feature after RoIAlign operation. It shows that the augmented feature becomes blurry compared with the original one, and such a domain gap causes detection performance degradation. Here we adopt the 1/8 input-resolution feature map from the R50 backbone trained with Deform-DETR w/ tck.

TABLE XIII  
ABLATION ON CROPPED FEATURE PROJECTOR IN FEATAUG-CROP ON  
DEFORM-DETR W/ TCK

Method	#ep.	AP	AP <sub>50</sub>	AP <sub>75</sub>	AP <sub>S</sub>	AP <sub>M</sub>	AP <sub>L</sub>
Deform-DETR w/ tck.	36	49.0	67.6	53.3	32.7	51.5	63.7
Same Proj.	24	49.2	67.8	53.6	31.8	52.9	63.4
Cropped Proj.	24	<b>49.6</b>	68.1	54.1	33.2	52.6	64.7

mislead the later Transformer decoder about the positional information of this image and thus harm the Transformer decoder training.

3) *Cropped Feature Projectors in FeatAug-DETR*: In our *FeatAug-Crop* and *FeatAug-FC*, in order to mitigate the domain gap caused by RoIAlign, we adopt projectors after the cropped features. The visualization of the caused domain gap is shown in Fig. 5. Here we ablate on removing the individual projectors and using the same projector on both the original and cropped features. The results are shown in Table XIII. Using the projectors for cropped features in *FeatAug-Crop* performs 0.4 AP better than using the same projector for both cropped and original features. The performance gain is more obvious on small objects and large objects, where  $AP_S$  increases by 1.4 and  $AP_L$  by 1.3.

### M. Visualization of Position-Aware Object Queries

In the pilot study in Section III-A, it shows that spatial augmentation changes the query-object assignments. We further visualize some bounding boxes that each object query predicts. Here we test a trained Deformable-DETR without our *DataAug-DETR* or *FeatAug-DETR*. We record all the output bounding box predictions that are matched with the ground truth objects for each query and visualize {center\_x, center\_y, width, height} of all the predicted bounding boxes for each query. Here we visualize four queries (#11, #27, #145, and #238) in Fig. 6.

As shown in Fig. 6, the predicted center of each object query is almost always located around a fixed position, which means that a spatial transformation that changes the bounding boxes' relative positions on the feature map (such as flipping) can change the matched object queries. Furthermore, by observing the predicted heights and widths distribution of Query #11 and Query #27, it shows that though the two queries predict similar center positions, their predicted height and width are differently distributed. Query #11 always predicts large objects

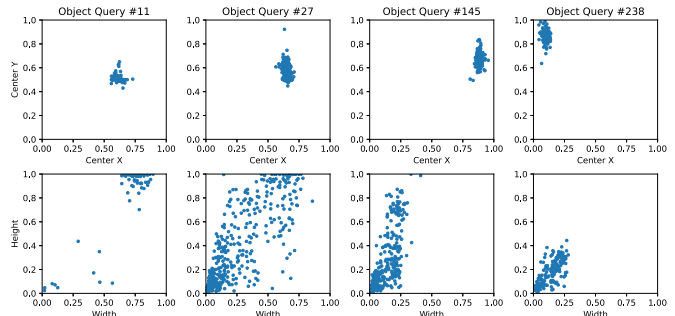


Fig. 6. We visualize the predicted bounding boxes position distribution of 4 randomly selected object queries on 10000 random selected images from COCO train2017 set. The top 4 figures show the predicted center x and center y of the corresponding bounding boxes, while the bottom 4 figures show the predicted heights and widths of the queries.

while Query #27 is always in charge of small objects. It shows that the cropping operation that changes the sizes of the predicted bounding boxes' also varies the object-query matchings.

The above two observations show that the object queries assigned to predicted bounding boxes are highly position-aware. Applying flipping and cropping operations makes different object queries match different ground truth objects.

## V. CONCLUSION

In this paper, we enrich the formulation of one-to-many matching for DETRs. We summarize the one-to-many matching mechanism of augmenting object queries for Group-DETR [8] and Hybrid Matching [9] and propose augmenting image features to implement one-to-many matching. To be detailed, we proposed *DataAug-DETR* method to help DETR-like methods to achieve higher performance after convergence. Further, we also propose the *FeatAug-DETR* method, including *FeatAug-Flip*, *FeatAug-Crop*, and *FeatAug-FC*. The *FeatAug-DETR* method significantly accelerates DETR training and accomplishes better performance. *FeatAug-DETR* improve the performance of Deformable-DETR [1] with different backbones for around 1.0 AP and shorten the training epochs to only 1× or 2× standard training schedule. When applying *FeatAug-DETR* together with Hybrid Matching [9] and using a Swin-Large backbone, we achieve the current state-of-the-art performance of 58.3 AP.

## REFERENCES

- [1] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," 2020, *arXiv: 2010.04159*.
- [2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.
- [3] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7263–7271.
- [4] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv: 1804.02767*.
- [5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.
- [6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

- [7] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.
- [8] Q. Chen, X. Chen, G. Zeng, and J. Wang, "Group DETR: Fast training convergence with decoupled one-to-many label assignment," 2022, *arXiv:2207.13085*.
- [9] D. Jia et al., "DETRs with hybrid matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 19 702–19 712.
- [10] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 213–229.
- [11] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
- [12] X. Cao, P. Yuan, B. Feng, and K. Niu, "CF-DETR: Coarse-to-fine transformers for end-to-end object detection," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 185–193.
- [13] F. Li, H. Zhang, S. Liu, J. Guo, L. M. Ni, and L. Zhang, "DN-DETR: Accelerate DETR training by introducing query denoising," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 13 619–13 627.
- [14] P. Gao, M. Zheng, X. Wang, J. Dai, and H. Li, "Fast convergence of DETR with spatially modulated co-attention," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 3621–3630.
- [15] X. Dai, Y. Chen, J. Yang, P. Zhang, L. Yuan, and L. Zhang, "Dynamic DETR: End-to-end object detection with dynamic attention," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 2988–2997.
- [16] Y. Wang, X. Zhang, T. Yang, and J. Sun, "Anchor DETR: Query design for transformer-based detector," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 2567–2575.
- [17] D. Meng et al., "Conditional DETR for fast training convergence," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 3651–3660.
- [18] S. Liu et al., "DAB-DETR: Dynamic anchor boxes are better queries for DETR," 2022, *arXiv:2201.12329*.
- [19] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10 012–10 022.
- [20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.
- [21] W. Liu et al., "SSD: Single shot multibox detector," in *Proc. 14th Eur. Conf. Comput. Vis.*, Springer, 2016, pp. 21–37.
- [22] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9627–9636.
- [23] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9759–9768.
- [24] K. Kim and H. S. Lee, "Probabilistic anchor assignment with IoU prediction for object detection," in *Proc. 16th Eur. Conf. Comput. Vis.*, Springer, 2020, pp. 355–371.
- [25] Z. Ge, S. Liu, Z. Li, O. Yoshie, and J. Sun, "OTA: Optimal transport assignment for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 303–312.
- [26] Z. Sun, S. Cao, Y. Yang, and K. M. Kitani, "Rethinking transformer-based set prediction for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 3611–3620.
- [27] Z. Yao, J. Ai, B. Li, and C. Zhang, "Efficient DETR: Improving end-to-end object detector with dense prior," 2021, *arXiv:2104.01318*.
- [28] X. Dai et al., "Dynamic head: Unifying object detection heads with attentions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7373–7382.
- [29] J. Dai et al., "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 764–773.
- [30] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets v2: More deformable, better results," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9308–9316.
- [31] H. Zhang et al., "DINO: DETR with improved denoising anchor boxes for end-to-end object detection," 2022, *arXiv:2203.03605*.
- [32] G. Brasó, N. Kister, and L. Leal-Taixé, "The center of attention: Centerkeypoint grouping via attention for multi-person pose estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 11 853–11 863.
- [33] K. Li, S. Wang, X. Zhang, Y. Xu, W. Xu, and Z. Tu, "Pose recognition with cascade transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1944–1953.
- [34] J. Zhang et al., "Direct multi-view multi-person 3D pose estimation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 13 153–13 164.
- [35] Y. Li et al., "TokenPose: Learning keypoint tokens for human pose estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 11 313–11 322.
- [36] L. Stoffl, M. Vidal, and A. Mathis, "End-to-end trainable multi-instance pose estimation with transformers," 2021, *arXiv:2103.12115*.
- [37] D. Shi, X. Wei, L. Li, Y. Ren, and W. Tan, "End-to-end multi-person pose estimation with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11 069–11 078.
- [38] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du, "BEVDet: High-performance multi-camera 3D object detection in bird-eye-view," 2021, *arXiv:2112.11790*.
- [39] J. Huang and G. Huang, "BEVDet4D: Exploit temporal cues in multi-camera 3D object detection," 2022, *arXiv:2203.17054*.
- [40] T. Liang et al., "BEVFusion: A simple and robust LiDAR-camera fusion framework," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 10 421–10 434.
- [41] I. Misra, R. Girdhar, and A. Joulin, "An end-to-end transformer model for 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 2906–2917.
- [42] X. Bai et al., "TransFusion: Robust LiDAR-camera fusion for 3D object detection with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 1090–1099.
- [43] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "DETR3D: 3D object detection from multi-view images via 3D-to-2D queries," in *Proc. Conf. Robot Learn.*, 2022, pp. 180–191.
- [44] Z. Li et al., "BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2022, pp. 1–18.
- [45] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5828–5839.
- [46] H. W. Kuhn, "The Hungarian Method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, no. 1/2, pp. 83–97, 1955.
- [47] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.
- [48] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. 13th Eur. Conf. Comput. Vis.*, Springer, 2014, pp. 740–755.
- [49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
- [51] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 658–666.
- [52] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
- [53] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv: 1711.05101*.
- [54] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2359–2367.



**Rongyao Fang** is currently working toward the PhD degree with Multi-Media Lab(MMLab), Chinese University of Hong Kong. His research interests center on unified models for visual perception and generation, leveraging representation learning, and visual perception model approaches. His focus on developing integrated systems that can perceive, understand, and generate visual content through advanced computer vision techniques.



**Peng Gao** received the PhD degree from Multimedia Lab, Chinese University of Hong Kong, in 2021. He is a Young scientist with Shanghai AI Lab. His research interests lie in multi-modality learning, efficient visual backbone design, and self-supervised representation learning.



**Si Liu** (Senior Member, IEEE) received the bachelor's degree from the Experimental Class of Beijing Institute of Technology (BIT), and the PhD degree from the Institute of Automation, Chinese Academy of Sciences (CASIA), under the supervision of Prof. Hanqing Lu. She is an associate professor with the Institute of Information Engineering, Chinese Academy of Sciences. She used to be a research fellow with Learning and Vision Research Group, Department of Electrical and Computer Engineering, National University of Singapore, directed by Prof. Shuicheng Yan. Her current research interests include object categorization, object detection, image parsing, and human pose estimation. She is also interested in the applications, such as makeup recommendation and synthesis, clothes retrieval, and clothes recommendation.



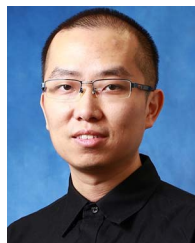
**Aojun Zhou** received the MS degree from the National Laboratory of Pattern Recognition (NLPR), Institute of Automation Chinese Academy of Science (CASIA), China. He is currently working toward the PhD degree with the Department of Electronic Engineering, Chinese University of Hong Kong. His research interests include deep learning and computer vision.



**Jifeng Dai** (Member, IEEE) is an associate professor with the Department of Electronic Engineering, Tsinghua University. His current research focus is on deep learning for high-level vision. Prior to that, he was an executive research director with SenseTime Research, headed by Professor Xiaogang Wang, between 2019 and 2022. He was a principle research manager with Visual Computing Group, Microsoft Research Asia (MSRA) between 2014 and 2019, headed by Dr. Jian Sun.



**Yingjie Cai** received the bachelor's degree from the CSE Department, Beihang University, in 2018. She is currently working toward the PhD degree with Multimedia Lab(MMLab), CUHK, supervised by Professor Hongsheng Li and Professor Xiaogang Wang. Her research interests include but not limited to 3D vision, especially on 3D perception, restoration and generation.



**Hongsheng Li** is an associate professor with the Department of Electronic Engineering, Chinese University of Hong Kong and Co-affiliated to Multimedia Laboratory.