DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection

Hao Zhang[1,3*†], Feng Li[1,3*†], Shilong Liu[2,3*†], Lei Zhang[3‡],
Hang Su[2], Jun Zhu[2], Lionel M. Ni[1,4], Heung-Yeung Shum[1,3]

[1]The Hong Kong University of Science and Technology.
[2]Dept. of CST., BNRist Center, Institute for AI, Tsinghua University.
[3]International Digital Economy Academy (IDEA).
[4]The Hong Kong University of Science and Technology (Guangzhou).
{hzhangcx, fliay}@connect.ust.hk
{liusl20}@mails.tsinghua.edu.cn
{suhangss, dcszj}@mail.tsinghua.edu.cn
{ni, hshum}@ust.hk
{leizhang}@idea.edu.cn

Abstract. We present DINO (**D**ETR with **I**mproved de**N**oising anch**O**r boxes), a state-of-the-art end-to-end object detector. DINO improves over previous DETR-like models in performance and efficiency by using a contrastive way for denoising training, a mixed query selection method for anchor initialization, and a look forward twice scheme for box prediction. DINO achieves 49.4AP in 12 epochs and 51.3AP in 24 epochs on COCO with a ResNet-50 backbone and multi-scale features, yielding a significant improvement of **+6.0AP** and **+2.7AP**, respectively, compared to DN-DETR, the previous best DETR-like model. DINO scales well in both model size and data size. Without bells and whistles, after pre-training on the Objects365 dataset with a SwinL backbone, DINO obtains the best results on both COCO `val2017` (**63.2AP**) and `test-dev` (**63.3AP**). Compared to other models on the leaderboard, DINO significantly reduces its model size and pre-training data size while achieving better results. Our code will be available at https://github.com/IDEACVR/DINO.

Keywords: Object Detection; Detection Transformer; End-to-End Detector

1 Introduction

Object detection is a fundamental task in computer vision. Remarkable progress has been accomplished by classical convolution-based object detection algo-

---

* Equal contribution. Listing order is random.

† This work was done when Hao Zhang, Feng Li, and Shilong Liu were interns at IDEA.
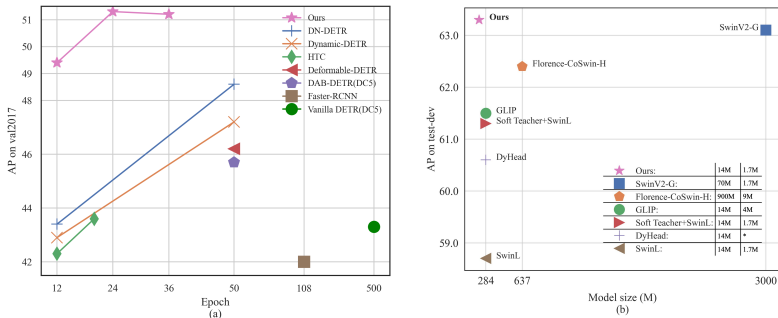
‡ Corresponding author.

Fig. 1. AP on COCO compared with other detection models. (a) Comparison to models with a ResNet-50 backbone w.r.t. training epochs. Models marked with DC5 use a dilated larger resolution feature map. Other models use multi-scale features. (b) Comparison to SOTA models w.r.t. pre-training data size and model size. SOTA models are from the COCO `test-dev` leaderboard. In the legend we list the backbone pre-training data size (first number) and detection pre-training data size (second number). ∗ means the data size is not disclosed.

rithms [31,35,19,2,12]. Despite that such algorithms normally include hand-designed components like anchor generation and non-maximum suppression (NMS), they yield the best detection models such as DyHead [7], Swin [23] and SwinV2 [22] with HTC++ [4], as evidenced on the COCO test-dev leaderboard [1].

In contrast to classical detection algorithms, DETR [3] is a novel Transformer-based detection algorithm. It eliminates the need of hand-designed components and achieves comparable performance with optimized classical detectors like Faster RCNN [31]. Different from previous detectors, DETR models object detection as a set prediction task and assigns labels by bipartite graph matching. It leverages learnable queries to probe the existence of objects and combine features from an image feature map, which behaves like soft ROI pooling [21].

Despite its promising performance, the training convergence of DETR is slow and the meaning of queries is unclear. To address such problems, many methods have been proposed, such as introducing deformable attention [41], decoupling positional and content information [25], providing spatial priors [11,39,37], etc. Recently, DAB-DETR [21] proposes to formulate DETR queries as dynamic anchor boxes (DAB), which bridges the gap between classical anchor-based detectors and DETR-like ones. DN-DETR [17] further solves the instability of bipartite matching by introducing a denoising (DN) technique. The combination of DAB and DN makes DETR-like models competitive with classical detectors on both training efficiency and inference performance.

The best detection models nowadays are based on improved classical detectors like DyHead [8] and HTC [4]. For example, the best result presented in SwinV2 [22] was trained with the HTC++ [4,23] framework. Two main reasons contribute to the phenomenon: 1) *Previous DETR-like models are inferior* to the improved classical detectors. Most classical detectors have been well studied and

highly optimized, leading to a better performance compared with the newly developed DETR-like models. For instance, the best performing DETR-like models nowadays are still under 50 AP on COCO. 2) The *scalability* of DETR-like models has not been well studied. There is no reported result about how DETR-like models perform when scaling to a large backbone and a large-scale data set. We aim to address both concerns in this paper.

Specifically, by improving the denoising training, query initialization, and box prediction, we design a new DETR-like model based on DN-DETR [17], DAB-DETR [21], and Deformable DETR [41]. We name our model as **DINO** (**D**ETR with **I**mproved de**N**oising anch**O**r box). As shown in Fig. 1, the comparison on COCO shows the superior performance of DINO. In particular, DINO demonstrates a great scalability, setting a new record of 63.3 AP on the COCO test-dev leaderboard [1]

As a DETR-like model, DINO contains a backbone, a multi-layer Transformer encoder, a multi-layer Transformer decoder, and multiple prediction heads. Following DAB-DETR [21], we formulate queries in decoder as dynamic anchor boxes and refine them step-by-step across decoder layers. Following DN-DETR [17], we add ground truth labels and boxes with noises into the Transformer decoder layers to help stabilize bipartite matching during training. We also adopt deformable attention [41] for its computational efficiency. Moreover, we propose three new methods as follows. First, to improve the one-to-one matching, we propose a *contrastive denoising training* by adding both positive and negative samples of the same ground truth at the same time. After adding two different noises to the same ground truth box, we mark the box with a smaller noise as positive and the other as negative. The contrastive denoising training helps the model to avoid duplicate outputs of the same target. Second, the dynamic anchor box formulation of queries links DETR-like models with classical two-stage models. Hence we propose a *mixed query selection* method, which helps better initialize the queries. We select initial anchor boxes as positional queries from the output of the encoder, similar to [41,39]. However, we leave the content queries learnable as before, encouraging the first decoder layer to focus on the spatial prior. Third, to leverage the refined box information from later layers to help optimize the parameters of their adjacent early layers, we propose a new *look forward twice* scheme to correct the updated parameters with gradients from later layers.

We validate the effectiveness of DINO with extensive experiments on the COCO [20] detection benchmarks. As shown in Fig. 1, DINO achieves 49.4AP in 12 epochs and 51.3AP in 24 epochs with ResNet-50 and multi-scale features, yielding a significant improvement of **+6.0**AP and **+2.7**AP, respectively, compared to the previous best DETR-like model. In addition, DINO scales well in both model size and data size. After pre-training on the Objects365 [33] data set with a SwinL [23] backbone, DINO achieves the best results on both COCO `val2017` (**63.2**AP) and `test-dev` (**63.3**AP) benchmarks, as shown in Table 3. Compared to other models on the leaderboard [1], we reduce the model size to **1/15** compared to SwinV2-G [22]. Compared to Florence [40], we reduce the

pre-training detection dataset to **1/5** and backbone pre-training dataset to **1/60** while achieving better results.

We summarize our contributions as follows.

1. We design a new end-to-end DETR-like object detector with several novel techniques, including contrastive DN training, mixed query selection, and look forward twice for different parts of the DINO model.
2. We conduct intensive ablation studies to validate the effectiveness of different design choices in DINO. As a result, DINO achieves 49.4AP in 12 epochs and 51.3AP in 24 epochs with ResNet-50 and multi-scale features, significantly outperforming the previous best DETR-like models. In particular, DINO trained in 12 epochs shows a more significant improvement on small objects, yielding an improvement of **+7.5**AP.
3. We show that, without bells and whistles, DINO can achieve the best performance on public benchmarks. After pre-training on the Objects365 [33] dataset with a SwinL [23] backbone, DINO achieves the best results on both COCO `val2017` (**63.2**AP) and `test-dev` (**63.3**AP) benchmarks. To the best of our knowledge, this is the first time that an end-to-end Transformer detector outperforms state-of-the-art (SOTA) models on the COCO leaderboard [1].

2   Related Work

2.1   Classical Object Detectors

Early convolution-based object detectors are either two-stage or one-stage models, based on hand-crafted anchors or reference points. Two-stage models [30,13] usually use an region proposal network (RPN) [30] to propose potential boxes, which are then refined in the second stage. One-stage models such as YOLO v2 [28] and YOLO v3 [29] directly output offsets relative to predefined anchors. Recently, some convolution-based models such as HTC++ [4] and Dyhead [7] have achieved top performance on the COCO 2017 dataset [20]. The performance of convolution-based models, however, relies on the way they generate anchors. Moreover, they need hand-designed components like NMS to remove duplicate boxes, and hence cannot perform end-to-end optimization.

2.2   DETR and Its Variants

Carion *et al.* [3] proposed a Transformer-based end-to-end object detector named DETR (DEtection TRansformer) without using hand-designed components like anchor design and NMS. Many follow-up papers have attempted to address the slow training convergence issue of DETR introduced by decoder cross-attention. For instance, Sun *et al.* [34] designed an encoder-only DETR without using a decoder. Dai *et al.* [7] proposed a dynamic decoder to focus on important regions from multiple feature levels.

Another line of works is towards a deeper understanding of decoder queries in DETR. Many papers associate queries with spatial position from different perspectives. Deformable DETR [41] predicts 2D anchor points and designs a deformable attention module that only attends to certain sampling points around a reference point. Efficient DETR [39] selects top K positions from encoder's dense prediction to enhance decoder queries. DAB-DETR [21] further extends 2D anchor points to 4D anchor box coordinates to represent queries and dynamically update boxes in each decoder layer. Recently, DN-DETR [17] introduces a denoising training method to speed up DETR training. It feeds noise-added ground-truth labels and boxes into the decoder and trains the model to reconstruct the original ones. Our work of DINO in this paper is based on DAB-DETR and DN-DETR, and also adopts deformable attention for its computational efficiency.

2.3   Large-scale Pre-training for Object Detection

Large-scale pre-training have had a big impact on both natural language processing [10] and computer vision [27]. The best performance detectors nowadays are mostly achieved with large backbones pre-trained on large-scale data. For example, Swin V2 [22] extends its backbone size to 3.0 Billion parameters and pre-trains its models with 70M privately collected images. Florence [40] first pre-trains its backbone with 900M privately curated image-text pairs and then pre-trains its detector with 9M images with annotated or pseudo boxes. In contrast, DINO achieves the SOTA result with a publicly available SwinL [23] backbone and a public dataset Objects365 [33] (1.7M annotated images) only.

3   DINO: DETR with Improved DeNoising Anchor Boxes

3.1   Preliminaries

As studied in Conditional DETR [25] and DAB-DETR [21], it becomes clear that queries in DETR [3] are formed by two parts: a positional part and a content part, which are referred to as positional queries and content queries in this paper. DAB-DETR [21] explicitly formulates each positional query in DETR as a 4D anchor box (x, y, w, h), where x and y are the center coordinates of the box and w and h correspond to its width and height. Such an explicit anchor box formulation makes it easy to dynamically refine anchor boxes layer by layer in the decoder.

DN-DETR [17] introduces a denoising (DN) training method to accelerate the training convergence of DETR-like models. It shows that the slow convergence problem in DETR is caused by the instability of bipartite matching. To mitigate this problem, DN-DETR proposes to additionally feed noised ground-truth (GT) labels and boxes into the Transformer decoder and train the model to reconstruct the ground-truth ones. The added noise $(\Delta x, \Delta y, \Delta w, \Delta h)$ is constrained by $|\Delta x| < \frac{\lambda w}{2}$, $|\Delta y| < \frac{\lambda h}{2}$, $|\Delta w| < \lambda w$, and $|\Delta y| < \lambda h$, where (x, y, w, h) denotes
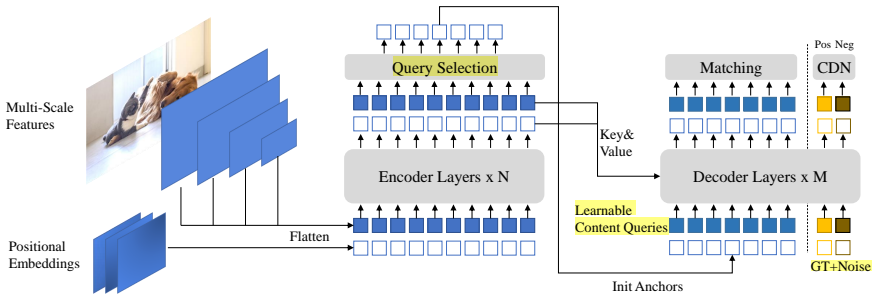
Fig. 2. The framework of our proposed DINO model. Our improvements are mainly in the Transformer encoder and decoder. The top-K encoder features in the last layer are selected to initialize the positional queries for the Transformer decoder, whereas the content queries are kept as learnable parameters. Our decoder also contains a Contrastive DeNoising (CDN) part with both positive and negative samples.

a GT box and λ^1 is a hyper-parameter to control the scale of noise. Since DN-DETR follows DAB-DETR to view decoder queries as anchors, a noised GT box can be viewed as a special anchor with a GT box nearby as λ is usually small. In addition to the orginal DETR queries, DN-DETR adds a DN part which feeds noised GT labels and boxes into the decoder to provide an auxiliary DN loss. The DN loss effectively stabilizes and speeds up the DETR training and can be plugged into any DETR-like models.

Deformable DETR [41] is another early work to speed up the convergence of DETR. To compute deformable attention, it introduces the concept of reference point so that deformable attention can attend to a small set of key sampling points around a reference. The reference point concept makes it possible to develop several techniques to further improve the DETR performance. The first technique is query selection[2], which selects features and reference boxes from the encoder as inputs to the decoder directly. The second technique is iterative bounding box refinement with a careful gradient detachment design between two decoder layers. We call this gradient detachment technique "look forward once" in our paper.

Following DAB-DETR and DN-DETR, DINO formulates the positional queries as dynamic anchor boxes and is trained with an extra DN loss. Note that DN-DETR also adopts several techniques from Deformable DETR to achieve a better performance, including its deformable attention mechanism and "look forward

---

[1] The DN-DETR paper [17] uses λ_1 and λ_2 to denote noise scales of center shifting and box scaling, but sets $\lambda_1 = \lambda_2$. In this paper, we use λ in place of λ_1 and λ_2 for simplicity.

[2] Also named as "two-stage" in the Deformable DETR paper. As the "two-stage" name might confuse readers with classical detectors, we use the term "query selection" instead in our paper.

once" implementation in layer parameter update. DINO further adopts the query selection idea from Deformable DETR to better initialize the positional queries. Built upon this strong baseline, DINO introduces three novel methods to further improve the detection performance, which will be described in Sec. 3.3, Sec. 3.4, and Sec. 3.5, respectively.

3.2   Model Overview

As a DETR-like model, DINO is an end-to-end architecture which contains a backbone, a multi-layer Transformer [36] encoder, a multi-layer Transformer decoder, and multiple prediction heads. The overall pipeline is shown in Fig. 2. Given an image, we extract multi-scale features with backbones like ResNet [14] or Swin Transformer [23], and then feed them into the Transformer encoder with corresponding positional embeddings. After feature enhancement with the encoder layers, we propose a new mixed query selection strategy to initialize anchors as positional queries for the decoder. Note that this strategy does not initialize content queries but leaves them learnable. More details of mixed query selection are available in Sec. 3.4. With the initialized anchors and the learnable content queries, we use the deformable attention [41] to combine the features of the encoder outputs and update the queries layer-by-layer. The final outputs are formed with refined anchor boxes and classification results predicted by refined content features. As in DN-DETR [17], we have an extra DN branch to perform denoising training. Beyond the standard DN method, we propose a new contrastive denoising training approach by taking into account hard negative samples, which will be presented in Sec. 3.3. To fully leverage the refined box information from later layers to help optimize the parameters of their adjacent early layer, a novel look forward twice method is proposed to pass gradients between adjacent layers, which will be described in Sec. 3.5.

3.3   Contrastive DeNoising Training

DN-DETR is very effective in stabilizing training and accelerating convergence. With the help of DN queries, it learns to make predictions based on anchors which have GT boxes nearby. However, it lacks a capability of predicting "no object" for anchors with no object nearby. To address this issue, we propose a Contrastive DeNoising (CDN) approach to *rejecting* useless anchors.

Implementation: DN-DETR has a hyper-parameter λ to control the noise scale. The generated noises are no larger than λ as DN-DETR wants the model to reconstruct the ground truth (GT) from moderately noised queries. In our method, we have two hyper-parameters λ_1 and λ_2, where $\lambda_1 < \lambda_2$. As shown in the concentric squares in Fig. 3, we generate two types of CDN queries: positive queries and negative queries. Positive queries within the inner square have a noise scale smaller than λ_1 and are expected to reconstruct their corresponding ground truth boxes. Negative queries between the inner and outer squares have a noise scale larger than λ_1 and smaller than λ_2. They are are expected to predict "no object". We usually adopt small λ_2 because hard negative samples closer to
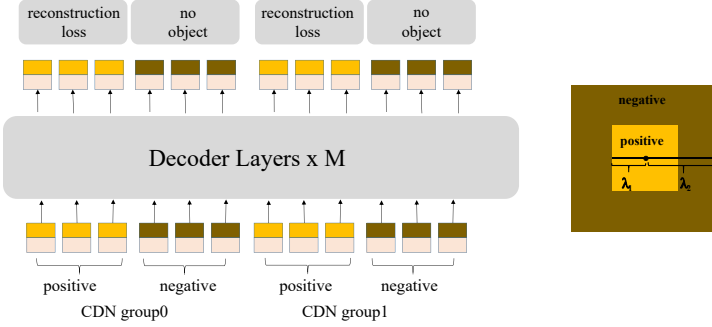
Fig. 3. The structure of CDN group and a demonstration of positive and negative examples. Although both positive and negative examples are 4D anchors that can be represented as points in 4D space, we illustrate them as points in 2D space on concentric squares for simplicity. Assuming the square center is a GT box, points inside the inner square are regarded as a positive example and points between the inner square and the outer square are viewed as negative examples.

GT boxes are more helpful to improve the performance. As shown in Fig. 3, each CDN group has a set of positive queries and negative queries. If an image has n GT boxes, a CDN group will have $2 \times n$ queries with each GT box generating a positive and a negative queries. Similar to DN-DETR, we also use multiple CDN groups to improve the effectiveness of our method. The reconstruction losses are l_1 and GIOU losses for box regression and focal loss [19] for classification. The loss to classify negative samples as background is also focal loss.

Analysis: The reason why our method works is that it can inhibit confusion and select high-quality anchors (queries) for predicting bounding boxes. The confusion happens when multiple anchors are close to one object. In this case, it is hard for the model to decide which anchor to choose. The confusion may lead to two problems. The first is duplicate predictions. Although DETR-like models can inhibit duplicate boxes with the help of set-based loss and self-attention [3], this ability is limited. As shown in the left figure of Fig. 8, when replacing our CDN queries with DN queries, the boy pointed by the arrow has 3 duplicate predictions. With CDN queries, our model can distinguish the slight difference between anchors and avoid duplicate predictions as shown in the right figure of Fig. 8. The second problem is that an unwanted anchor farther from a GT box might be selected. Although denoising training [17] has improved the model to choose nearby anchors, CDN further improves this capability by teaching the model to reject farther anchors.

Effectiveness: To demonstrate the effectiveness of CDN, we define a metric called Average Top-K Distance (ATD(k)) and use it to evaluate how far anchors are from their target GT boxes in the matching part. As in DETR, each anchor corresponds to a prediction which may be matched with a GT box or background. We only consider those matched with GT boxes here. Assume we have N GT bounding boxes $b_0, b_2, ..., b_{N-1}$ in a validation set, where $b_i = (x_i, y_i, w_i, h_i)$. For

each b_i, we can find its corresponding anchor and denote it as $a_i = (x_i', y_i', w_i', h_i')$. a_i is the initial anchor box of the decoder whose refined box after the last decoder layer is assigned to b_i during matching. Then we have

$$ATD(k) = \frac{1}{k} \sum \{topK\left(\{\|b_0 - a_0\|_1, \|b_1 - a_1\|_1, ..., \|b_{N-1} - a_{N-1}\|_1\}, k\right)\} \quad (1)$$

where $\|b_i - a_i\|_1$ is the l_1 distance between b_i and a_i and $topK(\mathbf{x}, k)$ is a function that returns the set of k largest elements in $\mathbf{x}$. The reason why we select the top-K elements is that the confusion problem is more likely to happen when the GT box is matched with a farther anchor. As shown in (a) and (b) of Fig. 4, DN is good enough for selecting a good anchor overall. However, CDN finds better anchors for small objects. Fig. 4 (c) shows that CDN queries lead to an improvement of +1.3 AP over DN queries on small objects in 12 epochs with ResNet-50 and multi-scale features.
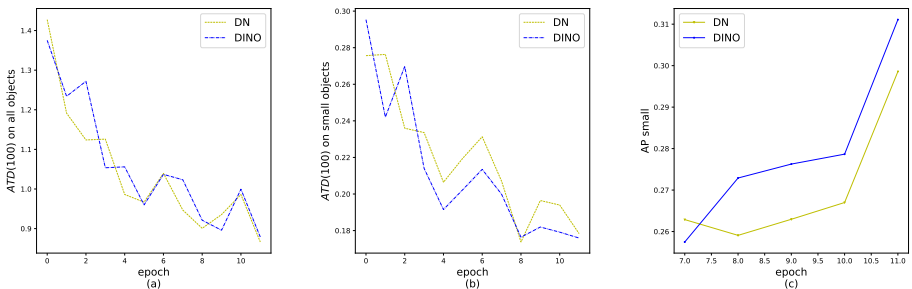

Fig. 4. (a) and (b) $ATD(100)$ on all objects and small objects respectively. (c) The AP on small objects.

3.4 Mixed Query Selection

In DETR [3] and DN-DETR [17], decoder queries are static embeddings without taking any encoder features from an individual image, as shown in Fig. 5 (a). They learn anchors (in DN-DETR and DAB-DETR) or positional queries (in DETR) from training data directly and set the content queries as all 0 vectors. Deformable DETR [41] learns both the positional and content queries, which is another implementation of static query initialization. To further improve the performance, Deformable DETR [41] has a query selection variant (called "two-stage" in [41]), which select top K encoder features from the last encoder layer as priors to enhance decoder queries. As shown in Fig. 5 (b), both the positional and content queries are generated by a linear transform of the selected features. In addition, these selected features are fed to an auxiliary detection head to get predicted boxes, which are used to initialize reference boxes. Similarly, Efficient DETR [39] also selects top K features based on the objectiveness (class) score of each encoder feature.
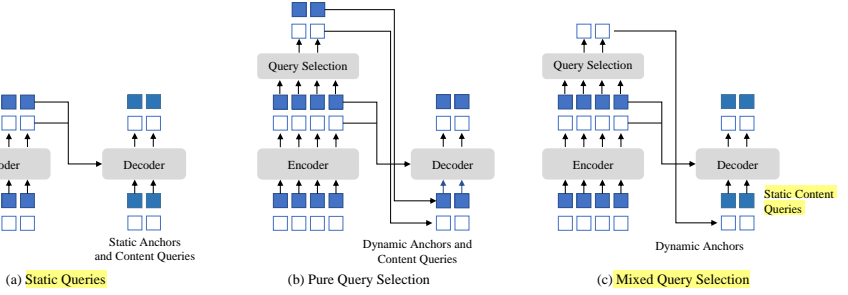
Fig. 5. Comparison of three different query initialization methods. The term "static" means that they will keep the same for different images in inference. A common implementation for these static queries is to make them learnable.

The dynamic 4D anchor box formulation of queries in our model makes it closely related to decoder positional queries, which can be improved by query selection. We follow the above practice and propose a mixed query selection approach. As shown in Fig. 5 (c), we only initialize anchor boxes using the position information associated with the selected top-K features, but leave the content queries static as before. Note that Deformable DETR [41] utilizes the top-K features to enhance not only the positional queries but also the content queries. As the selected features are preliminary content features without further refinement, they could be ambiguous and misleading to the decoder. For example, a selected feature may contain multiple objects or be only part of an object. In contrast, our mixed query selection approach only enhances the positional queries with top-K selected features and keeps the content queries learnable as before. It helps the model to use better positional information to pool more comprehensive content features from the encoder.

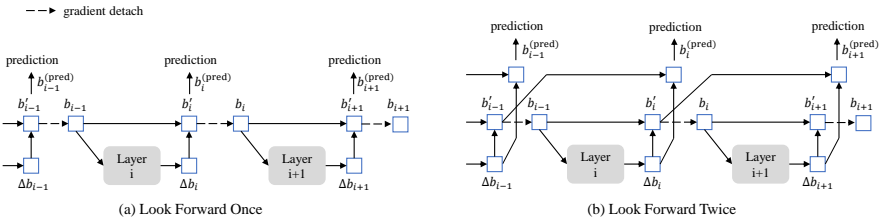## 3.5   Look Forward Twice

Fig. 6. Comparison of box update in Deformable DETR and our method.

We propose a new way to box prediction in this section. The iterative box refinement in Deformable DETR [41] blocks gradient back propagation to stabilize training. We name the method look forward once since the parameters of layer i are updated based on the auxiliary loss of boxes b_i only, as shown in Fig. 6 (a). However, we conjecture that the improved box information from a later layer could be more helpful to correct the box prediction in its adjacent early layer. Hence we propose another way called look forward twice to perform box update, where the parameters of layer-i are influenced by losses of both layer-i and layer-$(i+1)$, as shown in Fig. 6 (b). For each predicted offset Δb_i, it will be used to update box twice, one for b_i' and another for $b_{i+1}^{(pred)}$, hence we name our method as look forward twice.

The final precision of a predicted box $b_i^{(\text{pred})}$ is determined by two factors: the quality of the initial box b_{i-1} and the predicted offset of the box Δb_i. The look forward once scheme optimizes the latter only, as the gradient information is detached from layer-i to layer-$(i-1)$. In contrast, we improve both the initial box b_{i-1} and the predicted box offset Δb_i. A simple way to improving the quality is supervising the final box b_i' of layer i with the output of the next layer Δb_{i+1}. Hence we use the sum of b_i' and Δb_{i+1} as the predicted box of layer-$(i+1)$.

More specifically, given an input box b_{i-1} for the i-th layer, we obtain the final prediction box $b_i^{(\text{pred})}$ by:

$$\Delta b_i = \text{Layer}_i(b_{i-1}), \qquad b_i' = \text{Update}(b_{i-1}, \Delta b_i),$$
$$b_i = \text{Detach}(b_i'), \qquad b_i^{(\text{pred})} = \text{Update}(b_{i-1}', \Delta b_i), \qquad (2)$$

where b_i' is the undetached version of b_i. The term $\text{Update}(\cdot, \cdot)$ is a function that refines the box b_{i-1} by the predicted box offset Δb_i. We adopt the same way for box update[3] as in Deformable DETR [41].

4   Experiments

4.1   Setup

Dataset and Backbone: We conduct evaluation on the COCO 2017 object detection dataset [20], which is split into `train2017` and `val2017` (also called `minival`). We report results with two different backbones: ResNet-50 [14] pre-trained on ImageNet-1k [9] and SwinL [23] pre-trained on ImageNet-22k [9]. DINO with ResNet-50 is trained on `train2017` without extra data, while DINO with SwinL is first pre-trained on Object365 [33] and then fine-tuned on `train2017`. We report the standard average precision (AP) result on `val2017` under different IoU thresholds and object scales. We also report the `test-dev` results for DINO with SwinL.

---

[3] We use normalized forms of boxes in our model, hence each value of a box is a float between 0 and 1. Given two boxes, we sum them after inverse sigmoid and then transform the summation by sigmoid. Refer to Deformable DETR [41] Sec. A.3 for more details.

Implementation Details: DINO is composed of a backbone, a Transformer encoder, a Transformer decoder, and multiple prediction heads. In appendix D, we provide more implementation details, including all the hyper-parameters and engineering techniques used in our models for those who want to reproduce our results. We will release the code after the blind review.

4.2   Main Results

Model	Epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	GFLOPS	Params	FPS
Faster-RCNN(5scale) [30]	12	37.9	58.8	41.1	22.4	41.1	49.1	207	40M	21*
DETR(DC5) [3]	12	15.5	29.4	14.5	4.3	15.1	26.7	225	41M	20
Deformable DETR(4scale)[41]	12	41.1	–	–	–	–		196	40M	24
DAB-DETR(DC5)† [21]	12	38.0	60.3	39.8	19.2	40.9	55.4	256	44M	17
Dynamic DETR(5scale) [8]	12	42.9	61.0	46.3	24.6	44.9	54.4	–	58M	–
Dynamic Head(5scale) [7]	12	43.0	60.7	46.8	24.7	46.4	53.9	–	–	–
HTC(5scale) [4]	12	42.3	–	–	–	–		441	80M	5*
DN-Deformable-DETR(4scale)† [17]	12	43.4	61.9	47.2	24.8	46.8	59.4	265	48M	23
DINO-4scale†	12	**49.0**(+5.6)	**66.6**	**53.5**	**32.0**(+7.2)	**52.3**	**63.0**	279	47M	24
DINO-5scale†	12	**49.4**(+6.0)	**66.9**	**53.8**	**32.3**(+7.5)	**52.5**	**63.9**	860	47M	10

Table 1. Results for DINO and other detection models with the ResNet50 backbone on COCO `val2017` trained with 12 epochs (the so called 1× setting). For models without multi-scale features, we test their GFLOPS and FPS for their best model ResNet-50-DC5. DINO uses 900 queries. † indicates models that use 900 queries or 300 queries with 3 patterns which has similar effect with 900 queries. Other DETR-like models except DETR (100 queries) uses 300 queries. * indicates that they are tested using the mmdetection [5] framework.

12-epoch setting: With our improved anchor box denoising and training losses, the training process can be significantly accelerated. As shown in Table 1, we compare our method with strong baselines including both convolution-based methods [30,4,7] and DETR-like methods [3,41,8,21,17]. For a fair comparison, we report both GFLOPS and FPS tested on the same A100 NVIDIA GPU for all the models listed in Table 1. All methods except for DETR and DAB-DETR use multi-scale features. For those without multi-scale features, we report their results with ResNet-DC5 which has a better performance for its use of a dilated larger resolution feature map. Since some methods adopt 5 scales of feature maps and some adopt 4, we report our results with both 4 and 5 scales of feature maps.

As shown in Table 1, our method yields an improvement of +5.6 AP under the same setting using ResNet-50 with 4-scale feature maps and +6.0 AP with 5-scale feature maps. Our 4-scale model does not introduce much overhead in computation and the number of parameters. Moreover, our method performs especially well for small objects, gaining +7.2 AP with 4 scales and +7.5 AP with 5 scales. Note that the results of our models with ResNet-50 backbone are higher than those in the first version of our paper due to engineering techniques. **Comparison with the best models with a ResNet-50 backbone:**   To

Model	Epochs	AP	AP$_{50}$	AP$_{75}$	AP$_S$	AP$_M$	AP$_L$
Faster-RCNN [30]	108	42.0	62.4	44.2	20.5	45.8	61.1
DETR(DC5) [41]	500	43.3	63.1	45.9	22.5	47.3	61.1
Deformable DETR [41]	50	46.2	65.2	50.0	28.8	49.2	61.7
SMCA-R [11]	50	43.7	63.6	47.2	24.2	47.0	60.4
TSP-RCNN-R [34]	96	45.0	64.5	49.6	29.7	47.7	58.0
Dynamic DETR(5scale) [7]	50	47.2	65.9	51.1	28.6	49.3	59.1
DAB-Deformable-DETR [21]	50	46.9	66.0	50.8	30.1	50.4	62.5
DN-Deformable-DETR [17]	50	48.6	67.4	52.7	31.0	52.0	63.7
DINO-4scale	24	**50.4**(+1.8)	68.3	54.8	33.3	53.7	64.8
DINO-5scale	24	**51.3**(+2.7)	69.1	56.0	34.5	54.2	65.8
DINO-4scale	36	**50.9**(+2.3)	69.0	55.3	34.6	54.1	64.6
DINO-5scale	36	**51.2**(+2.6)	69.0	55.8	35.0	54.3	65.3

Table 2. Results for DINO and other detection models with the ResNet-50 backbone on COCO `val2017` trained with more epochs (24, 36, or more).
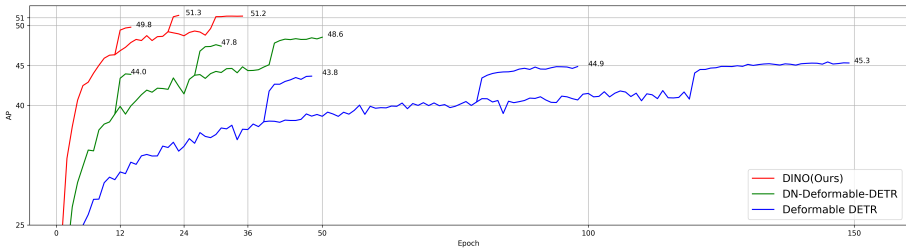

Fig. 7. Training convergence curves evaluated on COCO `val2017` for DINO and two previous state-of-the-art models with ResNet-50 using multi-scale features.

validate the effectiveness of our method in improving both convergence speed and performance, we compare our method with several strong baselines using the same ResNet-50 backbone. Despite the most common 50-epoch setting, we adopt the 24 (2×) and 36 (3×) epoch settings since our method converges faster and yields only a smaller additional gain with 50-epoch training. The results in Table 2 show that, using only 24 epochs, our method achieves an improvement of +1.8 AP and +2.7 AP with 4 and 5 scales, respectively. Moreover, using 36 epochs in the 3× setting, the improvement increases to +2.3 and +2.6 AP with 4 and 5 scales, respectively. The detailed convergence curve comparison is shown in Fig. 7.

Method	Params	Backbone Pre-training Dataset	Detection Pre-training Dataset	Use Mask	End-to-end	val2017 (AP)		test-dev (AP)	
						w/o TTA	w/ TTA	w/o TTA	w/ TTA
SwinL [23]	284M	IN-22K-14M	O365	✓		57.1	58.0	57.7	58.7
DyHead [7]	≥ 284M	IN-22K-14M	Unknown*			–	58.4	–	60.6
Soft Teacher+SwinL [38]	284M	IN-22K-14M	O365	✓		60.1	60.7	–	61.3
GLIP [18]	≥ 284M	IN-22K-14M	FourODs [18],GoldG+ [18,15]			–	60.8	–	61.5
Florence-CoSwin-H[40]	≥ 637M	FLD-900M [40]	FLD-9M [40]			–	62.0	–	62.4
SwinV2-G [22]	3.0B	IN-22K-ext-70M [22]	O365	✓		61.9	62.5	–	63.1
DINO-SwinL(Ours)	**218M**	IN-22K-14M	O365		✓	**63.1**	**63.2**	**63.2**	**63.3**

Table 3. Comparison of the best detection models on MS-COCO. Similar to DETR [3], we use the term "end-to-end" to indicate if a model is free from hand-crafted components like RPN and NMS. The term "use mask" means whether a model is trained with instance segmentation annotations. We use the terms "IN" and "O365" to denote the ImageNet [9] and Objects365 [33] datasets, respectively. Note that "O365" is a subset of "FourODs" and "FLD-9M". * DyHead does not disclose the details of the datasets used for model pre-training.

4.3   Comparison with SOTA Models

To compare with SOTA results, we use the publicly available SwinL [23] backbone pre-trained on ImageNet-22K. We first pre-train DINO on the Objects365 [33] dataset and then fine-tune it on COCO. As shown in Table 3, DINO achieves the best results of 63.2AP and 63.3AP on COCO `val2017` and `test-dev`, which demonstrate its strong scalability to larger model size and data size. Note that all the previous SOTA models in Table 3 do not use Transformer decoder-based detection heads (HTC++ [4] and DyHead [7]). It is the first time that an end-to-end Transformer detector is established as a SOTA model on the leaderboard [1]. Compared with the previous SOTA models, we use a much smaller model size (1/15 parameters compared with SwinV2-G [22]), backbone pre-training data size (1/60 images compared with Florence), and detection pre-training data size (1/5 images compared with Florence), while achieving better results. In addition, our reported performance without test time augmentation (TTA) is a neat result without bells and whistles. These results effectively show the superior detection performance of DINO compared with traditional detectors.

4.4   Ablation

Effectiveness of New Algorithm Components: To validate the effectiveness of our proposed methods, we build a strong baseline with optimized DN-DETR and pure query selection as described in section 3.1. We include all the pipeline optimization and engineering techniques (see section 4.1 and Appendix D) in the strong baseline. The result of the strong baseline is available in Table 4 Row 3. We also present the result of optimized DN-DETR without pure query selection from Deformable DETR [41] in Table 4 Row 2. While our strong baseline out-performs all previous models, our three new methods in DINO further improve the performance significantly.

#Row	QS	CDN	LFT	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1. DN-DETR [17]	No			43.4	61.9	47.2	24.8	46.8	59.4
2. Optimized DN-DETR	No			44.9	62.8	48.6	26.9	48.2	60.0
3. Strong baseline (Row2+pure query selection)	Pure			46.5	64.2	50.4	29.6	49.8	61.0
4. Row3+mixed query selection	Mixed			47.0	64.2	51.0	31.1	50.1	61.5
5. Row4+look forward twice	Mixed		✓	47.4	64.8	51.6	29.9	50.8	61.9
6. DINO (ours, Row5+contrastive DN)	Mixed	✓	✓	**47.9**	**65.3**	**52.1**	**31.2**	**50.9**	**61.9**

Table 4. Ablation comparison of the proposed algorithm components. We use the terms "QS", "CDN", and "LFT" to denote "Query Selection", "Contrastive De-Noising Training", and "Look Forward Twice", respectively.

5   Conclusion

In this paper, we have presented a strong end-to-end Transformer detector DINO with contrastive denoising training, mixed query selection, and look forward twice, which significantly improves both the training efficiency and the final detection performance. As a result, DINO outperforms all previous ResNet-50-based models on COCO `val2017` in both the 12-epoch and the 36-epoch settings using multi-scale features. Motivated by the improvement, we further explored to train DINO with a stronger backbone on a larger dataset and achieved a new state of the art, 63.3 AP on COCO 2017 `test-dev`. This result establishes DETR-like models as a mainstream detection framework, not only for its novel end-to-end detection optimization, but also for its superior performance.
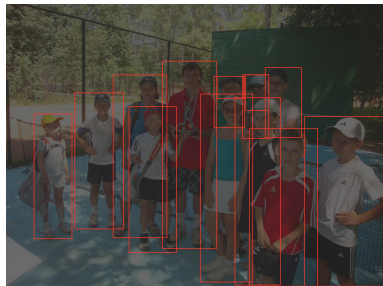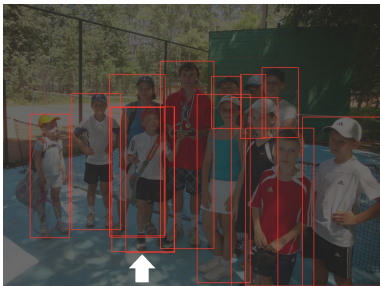

Fig. 8. The left figure is the detection result using a model trained with DN queries and the right is the result of our method. In the left image, the boy pointed by the arrow has 3 duplicate bounding boxes. For clarity, we only show boxes of class "person".

References

1. Papers with code - coco test-dev benchmark (object detection).
2. Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
3. Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.
4. Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019.
5. Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.
6. Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.
7. Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7373–7382, 2021.
8. Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2988–2997, October 2021.
9. 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*, pages 248–255. Ieee, 2009.
10. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pretraining of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
11. 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.
12. Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.
13. Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
14. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
15. Aishwarya Kamath, Mannat Singh, Yann LeCun, Ishan Misra, Gabriel Synnaeve, and Nicolas Carion. Mdetr – modulated detection for end-to-end multi-modal understanding. *arXiv: Computer Vision and Pattern Recognition*, 2021.
16. Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
17. Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dndetr: Accelerate detr training by introducing query denoising. *arXiv preprint arXiv:2203.01305*, 2022.

18. Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. *arXiv preprint arXiv:2112.03857*, 2021.
19. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020.
20. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
21. Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: Dynamic anchor boxes are better queries for DETR. *arXiv preprint arXiv:2201.12329*, 2022.
22. Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. *arXiv preprint arXiv:2111.09883*, 2021.
23. 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.
24. Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
25. 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.
26. Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2018.
27. Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.
28. Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.
29. Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
30. 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, 2015.
31. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.
32. Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019.
33. Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8430–8439, 2019.

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

35. Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9627–9636, 2019.

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

37. Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. *arXiv preprint arXiv:2109.07107*, 2021.

38. Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai, and Zicheng Liu. End-to-end semi-supervised object detection with soft teacher. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3060–3069, 2021.

39. 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.

40. Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

41. Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR 2021: The Ninth International Conference on Learning Representations*, 2021.

A   Test Time Augmentations (TTA)

We aim to build an end-to-end detector that is free from hand-crafted components. However, to compare with traditional detection models, we also explore the use of TTA in DETR-like models. We only use it in our large model with the SwinL backbone. Our TTA does not obtain an inspiring gain compared with traditional detectors, but we hope our exploration may provide some insights for future studies.

We adopt multi-scale test and horizontal flip as TTA. However, the way of ensembling different augmentations in our method is different from that in traditional methods which usually output duplicate boxes. In traditional methods, the ensembling is done by first gathering predictions from all augmentations and ranked by a confidence score. Then, duplicate boxes are found and eliminated by NMS or box voting. The reason why predictions from all augmentations are gathered first is that duplicate boxes appear not only among different augmentations but also within one augmentation. This ensembling method decreases the performance for our method since DETR-like methods are not prone to output duplicate boxes since their set-based prediction loss inhibits duplicate predictions and ensembling may incorrectly remove true positive predictions [3]. To address this issue, we designed a one-to-one ensembling method. Assume we have n augmentations $Aug_0, Aug_1, ..., Aug_{n-1}$, where Aug_i has predictions $\mathbf{O}^i$ and a pre-defined hyper-parameter weight w^i. $\mathbf{O}^i = \left\{ (b_0^i, l_0^i, s_0^i), (b_1^i, l_1^i, s_1^i), ..., (b_{m-1}^i, l_{m-1}^i, s_{m-1}^i) \right\}$ where b_j^i, l_j^i and s_j^i denote the j-th boundbox, label and score, respectively. We let Aug_0 be the main augmentation which is the most reliable one. For each prediction in $\mathbf{O}^0$, we select the prediction with the highest IOU from predictions of each of other augmentations $\mathbf{O}^1, ..., \mathbf{O}^{n-1}$ and make sure the IOU is higher than a predefined threshold. Finally, we ensemble the selected boxes through weighted average as follows

$$b = \frac{1}{\sum I^i} \sum_{i=o}^{n-1} I^i w^i s_{idx(i)}^i b_{idx(i)}^i \qquad (3)$$

where $I^i = 1$ when there is at least one box in $\mathbf{O}^i$ with IOU higher than the threshold and $I^i = 0$ otherwise. $idx(i)$ denotes the index of the selected box in $\mathbf{O}^i$.

B   Training Efficiency

We provide the GPU memory and training time for our base model in Table 5. All results are reported on 8 Nvidia A100 GPUs with ResNet-50 [14]. The results demonstrate that our models are not only effective but also efficient for training.

Model	Batch Size per GPU	Traning Time	GPU Mem.	Epoch	AP
Faster RCNN [30]*	8	~ 60min/ep	13GB	108	42.0
DETR [3]	8	~ 16min/ep	26GB	300	41.2
Deformable DETR [41]*	2	~ 55min/ep	16GB	50	45.4
DINO(Ours)	2	~ 55min/ep	16GB	**12**	**47.9**

Table 5. Training efficieny for different models with ResNet-50 backbone. All models are trianed with 8 Nvidia A100 GPUs. All results are reported by us. * The results of Faster RCNN are tested with the mmdetection framework. * We use the vanilla Deformable DETR without two-stage and bbox refinement during testing.

# Encoder/Decoder	6/6	4/6	3/6	2/6	6/4	6/2	2/4	2/2
AP	47.4	46.2	45.8	45.4	46.0	44.4	44.1	41.2

Table 6. Ablation on the numbers of encoder layers and decoder layers with the ResNet-50 backbone on COCO val2017. We use the 12-epoch setting and 100 DN queries without negative samples here.

C   Additional Analysis on our Model Components

Analysis on the Number of Encoder and Decoder Layers: We also investigate the influence of varying numbers of encoder and decoder layers. As shown in Table 6, decreasing the number of decoder layers hurts the performance more significantly. For example, using the same 6 encoder layers while decreasing the number of decoder layers from 6 to 2 leads to a 3.0 AP drop. This performance drop is expected as the boxes are dynamically updated and refined through each decoder layer to get the final results. Moreover, we also observe that compared with other DETR-like models like Dynamic DETR [7] whose performance drops by 13.8AP (29.1 vs 42.9) when decreasing the number of decoder layers to 2, the performance drop of DINO is much smaller. This is because our mixed query selection approach feeds the selected boxes from the encoder to enhance the decoder queries. Therefore, the decoder queries are well initialized and not deeply coupled with decoder layer refinement.

# Denoising query	100 CDN	1000 DN	200 DN	100 DN	50 DN	10 DN	No DN
AP	47.9	47.6	47.4	47.4	46.7	46.0	45.1

Table 7. Ablation on number of denoising queries with the ResNet-50 backbone on COCO validation. Note that 100 CND query pairs contains 200 queries which are 100 positive and 100 negative queries.

Analysis on Query Denoising: We continue to investigate the influence of

query denoising by varying the number of denoising queries. We use the optimized dynamic denoising group (detailed in Appendix D.1). As shown in Table 7, when we use less than 100 denoising queries, increasing the number can lead to a significant performance improvement. However, continuing to increase the DN number after 100 yields only a small additional or even worse performance improvement. We also analysis the effect of the number of encoder and decoder Layers in Appendix C.

D More Implementation Details

D.1 Dynamic DN groups

In DN-DETR, all the GT objects (label+box) in one image are collected as one GT group for denoising. To improve the DN training efficiency, multiple noised versions of the GT group in an image are used during training. In DN-DETR, the number of groups is set to five or ten according to different model sizes. As DETR-like models adopt mini-batch training, the total number of DN queries for each image in one batch is padded to the largest one in the batch. Considering that the number of objects in one image in COCO dataset ranges from 1 to 80, this design is inefficient and results in excessive memory consumption. To address this problem, we propose to fix the number of DN queries and dynamically adjust the number of groups for each image according to its number of objects.

D.2 Large-Scale Model Pre-trianing

Objects365 [33] is a large-scale detection data set with over $1.7M$ annotated images for training and $80,000$ annotated images for validation. To use the data more efficiently, We select the first $5,000$ out of $80,000$ validation images as our validation set and add the others to training. We pre-train DINO on Objects365 for 26 epochs using 64 Nvidia A100 GPUs and fine-tune the model on COCO for 18 epochs using 16 Nvidia A100 GPUS. Each GPU has a local batch size of 1 image only. In the fine-tuning stage, we enlarge the image size to $1.5\times$ (i.e., with max size 1200×2000). This adds around 0.5 AP to the final result. To reduce the GPU memory usage, we leverage checkpointing [6] and mixed precision [26] during training. Moreover, we use 1000 DN queries for this large model.

D.3 Other Implementation Details

Basic hyper-parameters. For hyper-parameters, as in DN-DETR, we use a 6-layer Transformer encoder and a 6-layer Transformer decoder and 256 as the hidden feature dimension. We set the initial learning rate (lr) as 1×10^{-4} and adopt a simple lr scheduler, which drops lr at the 11-th, 20-th, and 30-th epoch by multiplying 0.1 for the 12, 24, and 36 epoch settings with RestNet50, respectively. We use the AdamW [16,24] optimizer with weight decay of 1×10^{-4} and train our model on Nvidia A100 GPUs with batch size 16. Since DN-DETR

[17] adopts 300 decoder queries and 3 patterns [37], we use $300 \times 3 = 900$ decoder queries with the same computation cost. Learning schedules of our DINO with SwinL are available in the appendix.

Loss function. We use the L1 loss and GIOU [32] loss for box regression and focal loss [19] with $\alpha = 0.25, \gamma = 2$ for classification. As in DETR [3], we add auxiliary losses after each decoder layer. Similar to Deformable DETR [41], we add extra intermediate losses after the query selection module, with the same components as for each decoder layer. We use the same loss coefficients as in DAB-DETR [21] and DN-DETR [17], that is, 1.0 for classification loss, 5.0 for L1 loss, and 2.0 for GIOU loss.

Detailed model components. We also optimize the detection pipeline used in DAB-DETR [21] and DN-DETR [17]. Following DN-Deformable-DETR [17], we use the same multi-scale approach as in Deformable DETR [41] and adopt the deformable attention. DN-DETR uses different prediction heads with unshared parameters in different decoder layers. In addition, we introduce dynamic denoising group to increase denoising training efficiency and alleviate memory overhead (see Appendix D.1). In this work, we find that using a shared prediction head will add additional performance improvement. This also leads to a reduction of about one million parameters. In addition, we find the conditional queries [25] used in DAB-DETR does not suit our model and we do not include them in our final model.

Training augmentation. We use the same random crop and scale augmentation during training following DETR [3]. For example, we randomly resize an input image with its shorter side between 480 and 800 pixels and its longer side at most 1333. For DINO with SwinL, we pre-train the model using the default setting, but finetune using $1.5\times$ larger scale (shorter side between 720 and 1200 pixels and longer side at most 2000 pixels) to compare with models on the leaderboard [1]. Without using any other tricks, we achieve the result of 63.1 on `val2017` and 63.2 on `test-dev` without test time augmentation (TTA) (see Appendix A), outperforming the previous state-of-the-art result 63.1 achieved by SwinV2 [22] with a much neater solution.

Multi-scale setting. For our 4-scale models, we extract features from stages 2, 3, and 4 of the backbone and add an extra feature by down-sampling the output of the stage 4. An additional feature map of the backbone stage 1 is used for our 5-scale models. For hyper-parameters, we set $\lambda_1 = 1.0$ and $\lambda_2 = 2.0$ and use 100 CDN pairs which contain 100 positive queries and 100 negative queries.

D.4   Detailed Hyper-parameters

We list the hyper-parameters for those who want to reproduce our results in Table 8.

Item	Value
lr	0.0001
lr_backbone	1e-05
weight_decay	0.0001
clip_max_norm	0.1
pe_temperature	20
enc_layers	6
dec_layers	6
dim_feedforward	2048
hidden_dim	256
dropout	0.0
nheads	8
num_queries	900
enc_n_points	4
dec_n_points	4
transformer_activation	"relu"
batch_norm_type	"FrozenBatchNorm2d"
set_cost_class	2.0
set_cost_bbox	5.0
set_cost_giou	2.0
cls_loss_coef	1.0
bbox_loss_coef	5.0
giou_loss_coef	2.0
focal_alpha	0.25
dn_box_noise_scale	0.4
dn_label_noise_ratio	0.5

Table 8. Hyper-parameters used in our models.