Dino

DINO (**D**ETR with **I**mproved de**N**oising anch**O**r boxes) is 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.

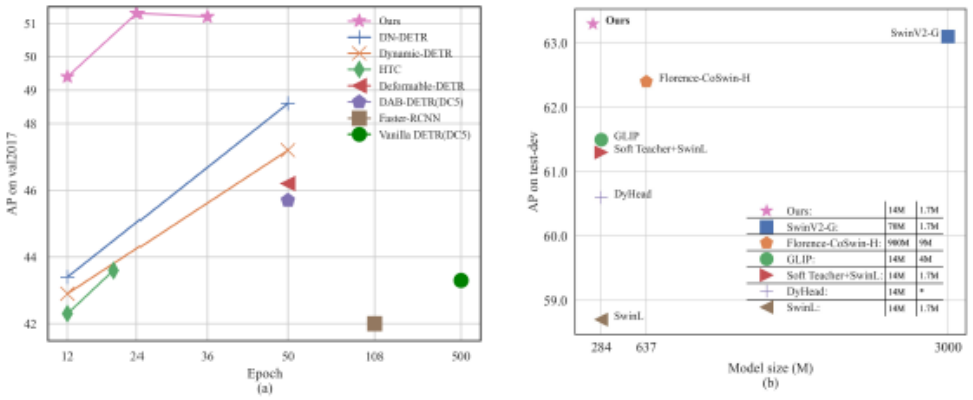

Fig. 1. AP on COCO compared with other detection models. (a) Comparison to models with a ResNet-50 backbone w.r.t. training epochs. Models marked with DC5 use a dilated larger resolution feature map. Other models use multi-scale features. (b) Comparison to SOTA models w.r.t. pre-training data size and 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.

DETR 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. 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.

Model structure:

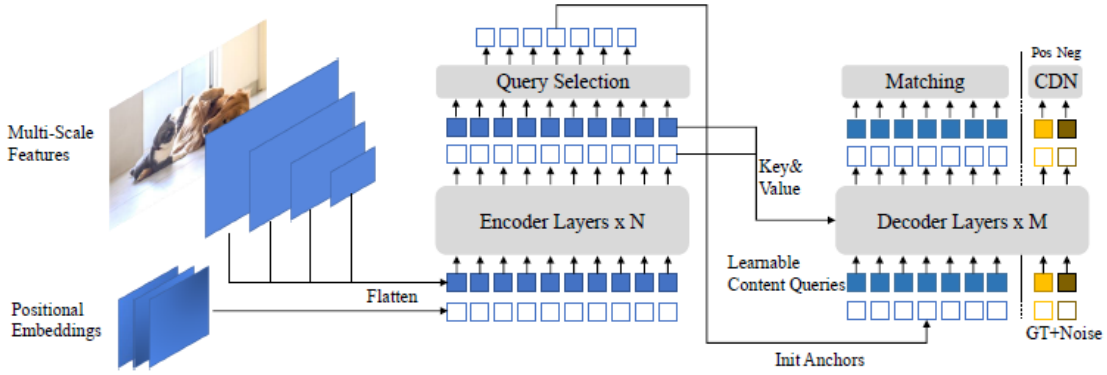

Fig. 2. The framework of our proposed DINO model. Our improvements are mainly in the Transformer encoder and decoder. The top-K encoder features in the last layer are selected to initialize the positional queries for the Transformer decoder, whereas the content queries are kept as learnable parameters. Our decoder also contains a Contrastive DeNoising (CDN) part with both positive and negative samples.

As a DETR-like model, DINO is an end-to-end architecture which contains a backbone, a multi-layer Transformer encoder, a multi-layer Transformer decoder, and multiple prediction heads. Given an image, we extract multi-scale features with backbones like ResNet or Swin Transformer, 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. With the initialized anchors and the learnable content queries, we use the deformable attention 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, 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. 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.

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.

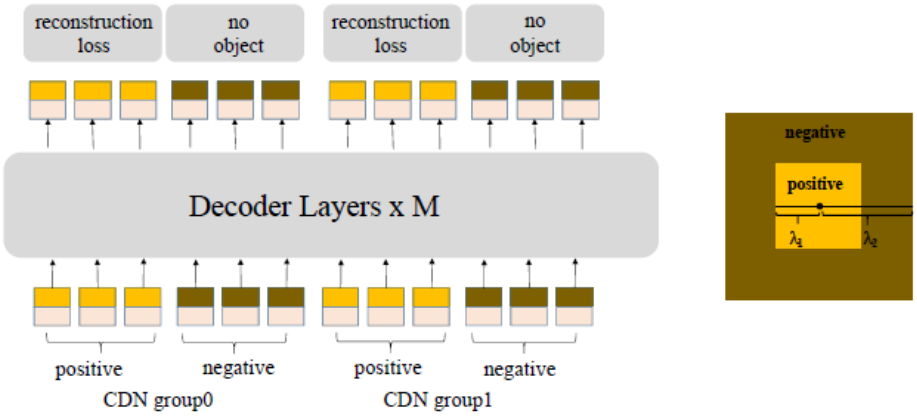

Fig. 3. The structure of CDN group and a demonstration of positive and negative examples. Although both positive and negative examples are 4D anchors that can be represented as points in 4D space, we illustrate them as points in 2D space on concentric squares for simplicity. Assuming the square center 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.

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 λ1 and λ2, where λ1 < λ2.

we generate two types of CDN queries: positive queries and negative queries. Positive queries within the inner square have a noise scale smaller than λ1 and are expected to reconstruct their corresponding ground truth boxes. Negative queries between the inner and outer squares have a noise scale larger than λ1 and smaller than λ2. They are are expected to predict “no object”. We usually adopt small λ2 because hard negative samples closer to GT boxes are more helpful to improve the performance. 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×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 l1 and GIOU losses for box regression and focal loss for classification. The loss to classify negative samples as background is also focal loss.

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.

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

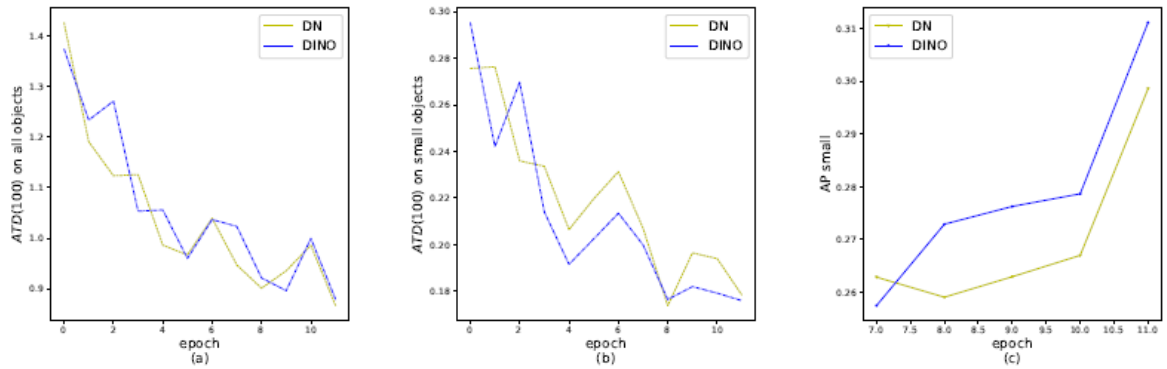

Fig. 4. (a) and (b) $ATD(100)$ on all objects and small objects respectively. (c) The AP on small objects.

Mixed Query Selection:

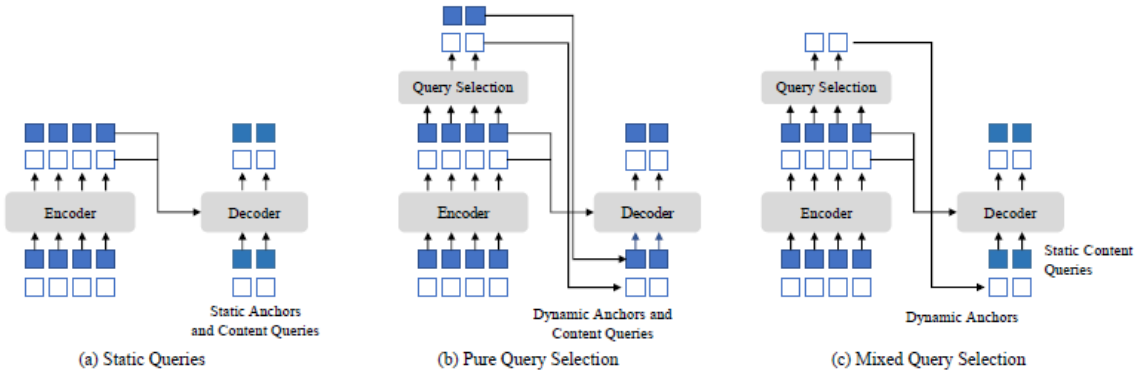

Fig. 5. Comparison of three different query initialization methods. The term "static" means that they will keep the same for different images in inference. A common implementation for these static queries is to make them learnable.

In DETR [3] and DN-DETR [17], decoder queries are static embeddings without taking any encoder features from an individual image. 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 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 ”twostage”), which select top K encoder features from the last encoder layer as priors to enhance decoder queries.

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 also selects top K features based on the objectiveness (class) score of each encoder feature.

we only initialize anchor boxes using the position information associated with the selected top-K features, but leave the content queries static as before.

Deformable DETR 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. 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.

Look Forward Twice:

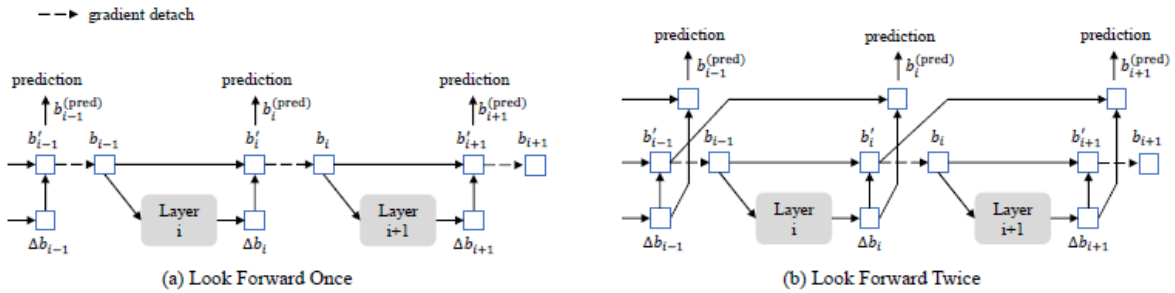

Fig. 6. Comparison of box update in Deformable DETR and our method.

Experiment:

Dataset: COCO 2017 object detection dataset

Results with two backbones: 1) ResNet-50 pretrained on ImageNet-1k

2) SwinL pre-trained on ImageNet-22k

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.

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).

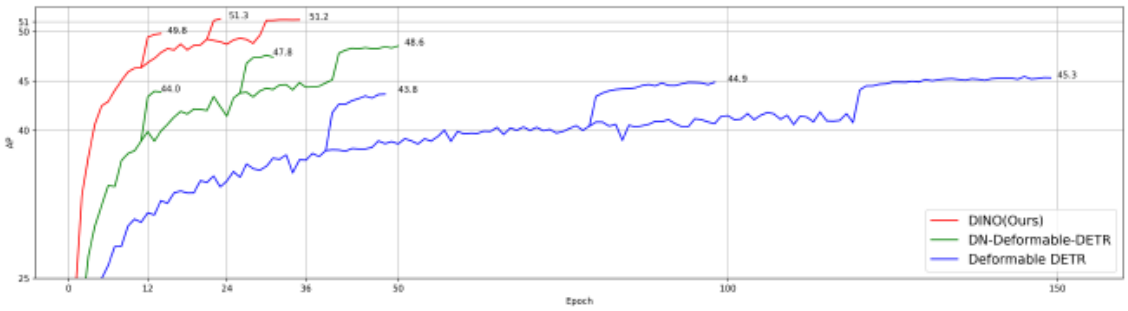

Fig. 7. Training convergence curves evaluated on COCO `val2017` for DINO and two previous state-of-the-art models with ResNet-50 using multi-scale features.

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.

#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.

# 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.

Hyperparameters:

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.