DN-DETR: Accelerate DETR Training by Introducing Query DeNoising

Feng Li\textsuperscript{1,2\dagger}, Hao Zhang\textsuperscript{1,2\dagger}, Shilong Liu\textsuperscript{2,3\dagger}, Jian Guo\textsuperscript{2}, Lionel M. Ni\textsuperscript{1,4}, Lei Zhang\textsuperscript{2\dagger}

\textsuperscript{1}The Hong Kong University of Science and Technology.
\textsuperscript{2}International Digital Economy Academy (IDEA).
\textsuperscript{3}Tsinghua University.
\textsuperscript{4}The Hong Kong University of Science and Technology (Guangzhou).

\{fliay, hzhangcx\}@connect.ust.hk
\{liusl20\}@mails.tsinghua.edu.cn
\{ni\}@ust.hk
\{guojian, leizhang\}@idea.edu.cn

Abstract

We present in this paper a novel denoising training method to speedup DETR (DEtection TRansformer) training and offer a deepened understanding of the slow convergence issue of DETR-like methods. We show that the slow convergence results from the instability of bipartite graph matching which causes inconsistent optimization goals in early training stages. To address this issue, except for the Hungarian loss, our method additionally feeds ground-truth bounding boxes with noises into Transformer decoder and trains the model to reconstruct the original boxes, which effectively reduces the bipartite graph matching difficulty and leads to a faster convergence. Our method is universal and can be easily plugged into any DETR-like methods by adding dozens of lines of code to achieve a remarkable improvement. As a result, our DN-DETR results in a remarkable improvement (+1.9 AP) under the same setting and achieves the best result (AP 43.4 and 48.6 with 12 and 50 epochs of training respectively) among DETR-like methods with ResNet-50 backbone. Compared with the baseline under the same setting, DN-DETR achieves comparable performance with 50\% training epochs. Code is available at https://github.com/FengLi-ust/DN-DETR.

1. Introduction

Object detection is a fundamental task in computer vision which aims to predict the bounding boxes and classes of objects in an image. While having made remarkable progress, classical detectors [14, 15] were mainly based on convolutional neural networks, until Carion et al. [1] recently introduced Transformers [17] into object detection and proposed DETR (DEtection TRansformer).

In contrast to previous detectors, DETR uses learnable queries to probe image features from the output of Transformer encoders and bipartite graph matching to perform set-based box prediction. Such a design effectively eliminates hand-designed anchors and non-maximum suppression (NMS) and makes object detection end-to-end optimizable. However, DETR suffers from prohibitively slow training convergence compared with previous detectors. To obtain a good performance, it usually takes 500 epochs of training on the COCO detection dataset, in contrast to 12 epochs used in the original Faster-RCNN training.

Much work [3, 11, 12, 16, 18, 20] has tried to identify the root cause and mitigate the slow convergence issue. Some of them address the problem through improving the model architecture. For example, Sun et al. [16] attribute the slow convergence issue to the low efficiency...
of the cross-attention and proposed an encoder-only DETR. 
Dai et al. [3] designed a ROI-based dynamic decoder to help 
the decoder focus on regions of interest. More recent works 
propose to associate each DETR query with a specific spa-
tial position rather than multiple positions for more efficient 
feature probing [11, 12, 18, 20]. For instance, Conditional 
DETR [12] decouples each query into a content part and a 
positional part, enforcing a query to have a clear correspon-
dence with a specific spatial position. Deformable DETR 
[20] and Anchor DETR [18] directly treat 2D reference 
points as queries to perform cross-attention. DAB-DETR 
[11] interprets queries as 4-D anchor boxes and learns to 
progressively improve them layer by layer.

Despite all the progress, few work pays attention to the 
bipartite graph matching part for more efficient training. In 
this study, we find that the slow convergence issue also re-
results from the discrete bipartite graph matching component, 
which is unstable especially in the early stages of training 
due to the nature of stochastic optimization. As a conse-
quence, for the same image, a query is often matched with 
different objects in different epochs, which makes optimization 
ambiguous and inconstant.

To address this problem, we propose a novel training 
method by introducing a query denoising task to help stabi-
lize bipartite graph matching in the training process. Since 
previous works have shown effective to interpret queries as 
contain positional information, we follow their viewpoint and 
use 4D anchor boxes as queries. Our solution is to feed 
noised ground truth bounding boxes as noised queries to-
gether with learnable anchor queries into Transformer de-
coders. Both kinds of queries have the same input format 
of \((x, y, w, h)\) and can be fed into Transformer decoders si-
multaneously. For noised queries, we perform a denoising 
task to reconstruct their corresponding ground truth boxes. 
For other learnable anchor queries, we use the same training 
loss including bipartite matching as in the vanilla DETR. 
As the noised bounding boxes do not need to go through 
the bipartite graph matching component, the denoising task 
can be regarded as an easier auxiliary task, helping DETR 
 alleviate the unstable discrete bipartite matching and learn 
bounding box prediction more quickly. Meanwhile, the de-
noising task also helps lower the optimization difficulty be-
cause the added random noise is usually small. To maxi-
imize the potential of this auxiliary task, we also regard each 
derived query as a bounding box + a class label embedding 
so that we are able to conduct both box denoising and label 
denoising.

In summary, our method is a denoising training ap-
proach. Our loss function consists of two components. One 
is a reconstruction loss and the other is a Hungarian loss 
which is the same as in other DETR-like methods. Our 
method can be easily plugged into any existing DETR-like 
method. For convenience, we utilize DAB-DETR [11] to 
evaluate our method since their decoder queries are explic-
titly formulated as 4D anchor boxes \((x, y, w, h)\). For DETR 
variants that only support 2D anchor points such as anchor 
DETR [18], we can do denoising on anchor points. For 
those that do not support anchors like the vanilla DETR [1], 
we can do linear transformation to map 4D anchor boxes to 
the same latent space as for other learnable queries.

To the best of our knowledge, this is the first work to 
introduce the denoising principle into detection models. We 
summarize our contribution as follows:

1. We design a novel training method to speedup DETR 
training. Experimental results show that our method 
not only accelerates training convergence, but also 
leads to a remarkably better training result — achieve 
the best result among all detection algorithms in the 
12-epoch setting. Moreover, our method shows a re-
markable improvement (+1.9 AP) over our baseline 
DAB-DETR and can be easily integrated into other 
DETR-like methods.

2. We analyze the slow convergence of DETR from a 
novel viewpoint and give a deeper understanding of 
DETR training. We design a metric to evaluate the 
instability of bipartite matching and verify that our 
method can effectively lower the instability.

3. We conduct a series of ablation studies to analyze the 
effectiveness of different components of our model 
such as noise, label embedding, and attention mask.

2. Related Work

Classical CNN-based detectors can be divided into 2 cat-
ergories, one-stage and two-stage methods. Two-stage meth-
ods [6, 7] first generate some region proposals and then de-
cide whether each region contains an object and do bound-
ing box regression to get a refined box. Ren et al. [15] pro-
posed an end-to-end method which utilizes a Region Pro-
posal Network to predict anchor boxes. In contrast to two-
stage methods, one-stage methods [13, 14] directly predict 
the offset of real boxes relative to anchor boxes. Overall, 
they are all anchor-based methods.

Carion et al. [1] proposed an end-to-end object detector 
based on Transformers [17] named DETR (DEtect-
ion TRansformer) without using anchors. While DETR 
achieves comparable results with Faster-RCNN [15], its 
training suffers severely from the slow convergence prob-
lem — it needs 500 epochs of training to obtain a good per-
formance.

Many recent works have attempted to speedup the train-
ing process of DETR. Some find the cross attention of 
Transformer decoders in DETR inefficient and make im-
provement from different ways. For example, Dai et al. [3]
designed a dynamic decoder that can focus on regions of interests from a coarse-to-fine manner and lower the learning difficulty. Sun et al. [16] discarded the Transformer decoder and proposed an encoder only DETR. Another series of works make improvements in decoder queries. Zhu et al. [20] designed an attention module that only attend to some sampling points around a reference point. Meng et al. [12] decoupled each decoder query into a content part and a position part and only utilized the content-to-content and position-to-position terms in the cross-attention formulation. Yao et al. [19] utilized a Region Proposal Network (RPN) to propose top-$K$ anchor points. DAB-DETR [11] uses 4-D box coordinates as queries and updates boxes layer-by-layer in a cascade manner.

Despite all the progress, none of them treats bipartite graph matching used in the Hungarian loss as a main reason for slow convergence. Sun et al. [16] analyzed the impact of Hungarian loss by using a pre-trained DETR as a teacher to provide the ground-truth label assignment for a student model and train the student model. They found that the label assignment only helps the convergence in the early stage of training but does not influence the final performance significantly. Therefore, they concluded that the Hungarian loss is not a main reason for slow convergence. In this work, we give a different analysis with an effective solution that leads to a different conclusion.

We adopt DAB-DETR as the detection architecture to evaluate our training method, where the label embedding appended with an indicator is used to replace the decoder embedding part to support label denoising. The difference between our method and other methods is mainly in the training method. In addition to the Hungarian loss, we add a denoising loss as an easier auxiliary task that can accelerate training and boost the performance significantly. Chen et al. [12] decoupled each decoder query into a content and a position part and only utilized the content-to-content and position-to-position terms in the cross-attention formulation. Zhu et al. [12] utilized a Region Proposal Network (RPN) to propose top-$K$ anchor points. DAB-DETR [11] uses 4-D box coordinates as queries and updates boxes layer-by-layer in a cascade manner.

Despite all the progress, none of them treats bipartite graph matching used in the Hungarian loss as a main reason for slow convergence. Sun et al. [16] analyzed the impact of Hungarian loss by using a pre-trained DETR as a teacher to provide the ground-truth label assignment for a student model and train the student model. They found that the label assignment only helps the convergence in the early stage of training but does not influence the final performance significantly. Therefore, they concluded that the Hungarian loss is not a main reason for slow convergence. In this work, we give a different analysis with an effective solution that leads to a different conclusion.

3. Why Denoising accelerates DETR training?

Hungarian matching is a popular algorithm in graph matching. Given a cost matrix, the algorithm outputs an optimal matching result. DETR is the first algorithm that adopts Hungarian matching in object detection to solve the matching problem between predicted objects and ground truth objects. DETR turns ground truth assignment to a dynamic process, which brings in an instability problem due to its discrete bipartite matching and the stochastic training process. There are works [5] showing that Hungarian matching does not result in a stable matching since blocking pairs exist. A small change of the cost matrix may cause an enormous change in the matching result, which will further lead to inconsistent optimization goals for decoder queries.

We view the training process of DETR-like models as two stages, learning “good anchors” and learning relative offsets. Decoder queries are responsible for learning anchors as shown in previous works [11,20]. The inconsistent update of anchors can make it difficult to learn relative offsets. Therefore, in our method, we leverage a denoising task as a training shortcut to make relative offset learning easier, as the denoising task bypasses bipartite matching. Since we interpret each decoder query as a 4-D anchor boxes, a noised query can be regarded as a “good anchor” which has a corresponding ground truth box nearby. The denoising training thus has a clear optimization goal - to predict the original bounding box, which essentially avoids the ambiguity brought by Hungarian matching.

To quantitatively evaluate the instability of the bipartite matching result, we design a metric as follows. For a training image, we denote the predicted objects from Transformer decoders as $O^i = \{O_0^i, O_1^i, ..., O_{N-1}^i\}$ in the $i$-th epoch, where $N$ is the number of predicted objects, and the ground truth objects as $T = \{T_0, T_1, T_2, ..., T_{M-1}\}$ where $M$ is the number of ground truth objects. After bipartite matching, we compute an index vector $V^i = \{V_0^i, V_1^i, ..., V_{N-1}^i\}$ to store the matching result of epoch $i$ as follows.

$$V^i_n = \begin{cases} m, & \text{if } O_n^i \text{ matches } T_m \\ -1, & \text{if } O_n^i \text{ matches nothing} \end{cases}$$

(1)

We define the instability of epoch $i$ for one training image as the difference between its $V^i$ and $V^{i-1}$, which is calculated as

$$IS^i = \sum_{j=0}^{N} I(V_n^i \neq V_n^{i-1})$$

(2)
where $\mathbb{1}(\cdot)$ is the indicator function. $\mathbb{1}(x) = 1$ if $x$ is true and 0 otherwise. The instability of epoch $i$ for the whole data set is averaged over the instability numbers for all images. We omit the index for an image for notation simplicity in Eq. (1) and Eq. (2).

Fig. 2 shows a comparison of IS between our DN-DETR (DeNoising DETR) and DAB-DETR. We conduct this evaluation on COCO 2017 validation set [10] which has 7.36 objects per image on average. So the largest possible IS is $7.36 \times 2 = 14.72$. Fig. 2 clearly shows that our method effectively alleviates the instability of matching.

4. DN-DETR

![Diagram](image_url)

Figure 3. Comparison of the cross-attention part DAB-DETR and our DN-DETR (a) DAB-DETR directly uses dynamically updated anchor boxes to provide both a reference query point $(x, y)$ and a reference anchor size $(w, h)$ to improve the cross-attention computation. (b) DN-DETR specify the decoder embedding as label embedding and add an indicator to differentiate denoising task and matching task.

4.1. Overview

We base on the architecture of DAB-DETR [11] to implement our training method. Similar to DAB-DETR, we explicitly formulates the decoder queries as box coordinates. The only difference between our architecture and theirs lies in the decoder embedding, which is specified as class label embedding to support label denoising. Our main contribution is the training method as shown in Fig. 4.

![Diagram](image_url)

Similar to DETR, our architecture contains a Transformer encoder and a Transformer decoder. On the encoder side, the image features are extracted with a CNN backbone and then fed into the Transformer encoder with positional encodings to attain refined image features. On the decoder side, queries are fed into the decoder to search for objects through cross attention.

We denote decoder queries as $\mathbf{q} = \{q_0, q_1, \ldots, q_{N-1}\}$ and the output of the Transformer decoder as $\mathbf{o} = \{o_0, o_1, \ldots, o_{N-1}\}$. We also use $F$ and $A$ to denote the refined image features after the Transformer encoder and the attention mask derived based on the denoising task design.

We can formulate our method as follows.

$$\mathbf{o} = D(\mathbf{q}, F | A) \quad (3)$$

where $D$ denotes the Transformer decoder.

There are two parts of decoder queries. One is the matching part. The inputs of this part are learnable anchors, which are treated in the same way as in DETR. That is, the matching part adopts bipartite graph matching and learns to approximate the ground truth box-label pairs with matched decoder outputs. The other is the denoising part. The inputs of this part are noised ground-truth (GT) box-label pairs which are called GT objects in rest of the paper. The outputs of the denoising part aims to reconstruct GT objects.

In the following, we abuse the notations to denote the denoising part as $\mathbf{q} = \{q_0, q_1, \ldots, q_{K-1}\}$ and the matching part as $\mathbf{Q} = \{Q_0, Q_1, \ldots, Q_{L-1}\}$. So the formulation of our method becomes

$$\mathbf{o} = D(\mathbf{q}, \mathbf{Q}, F | A) \quad (4)$$

To increase the denoising efficiency, we propose to use multiple versions of noised GT objects in the denoising part. Further more, we utilize an attention mask to prevent information leakage from the denoising part to the matching part and among different noised versions of the same GT object.

4.2. Intro to DAB-DETR

Many recent works associate DETR queries with different positional information. DAB-DETR follows this analysis and explicitly formulates each query as 4D anchor coordinates. As shown in Fig. 3(a), a query is specified as a tuple $(x, y, w, h)$, where $x, y$ are the center coordinates and $w, h$ are the corresponding width and height of each box. In addition, the anchor coordinates are dynamically updated layer by layer. The output of each decoder layer contains a tuple $(\Delta x, \Delta y, \Delta w, \Delta h)$ and the anchor is updated to $(x + \Delta x, y + \Delta y, w + \Delta w, h + \Delta h)$.

Note that our proposed method is mainly a training method which can be integrated into any DETR-like models. To test on DAB-DETR, we only add minimal modifications: specifying the decoder embedding as label embedding, as shown in Fig. 3(b).

4.3. Denoising

For each image, we collect all GT objects and add random noises to both their bounding boxes and class labels. To maximize the utility of denoising learning, we use multiple noised versions for each GT object.

We consider adding noise to boxes in two ways: center shifting and box scaling. We define $\lambda_1$ and $\lambda_2$ as the noise scale of these 2 noises. For center shifting, we add a random noise $(\Delta x, \Delta y)$, to the box center and make sure that $|\Delta x| < \frac{\lambda_1 w}{2}$ and $|\Delta y| < \frac{\lambda_1 h}{2}$, where $\lambda_1 \in (0, 1)$ so that the center of the noised box will still lie inside the
original bounding box. For box scaling, we set a hyperparameter $\lambda_2 \in (0, 1)$. The width and height of the box are randomly sampled in $[(1 - \lambda_2)w, (1 + \lambda_2)w]$ and $[(1 - \lambda_2)h, (1 + \lambda_2)h]$, respectively.

For label noise, we adopt label flipping, which means we randomly flip some ground-truth labels to other labels. Label flipping forces the model to predict the ground-truth labels according to the noised boxes to better capture label-box relationship. We have a hyper-parameter $\gamma$ to control the ratio of labels to flip. The reconstruction losses are $l_1$ loss and GIOU loss for boxes and focal loss [9] for class labels as in DAB-DETR. We use a function $\delta(\cdot)$ to denote the the noised GT objects. Therefore, each query in the denoising part can be represented as $q_k = \delta(t_m)$ where $t_m$ is $m$-th GT object.

Notice that denoising is only considered in training, during inference the denoising part is removed, leaving only the matching part.

4.4. Attention Mask

Attention mask is a component of great importance in our model. Without attention mask, the denoising training will compromise the performance instead of improving it as shown in Table 4.

To introduce attention mask, we need to first divide the noised GT objects into groups. Each group is a noised version of all GT objects. The denoising part becomes

$$q = \{g_0, g_1, \ldots, g_{P-1}\}$$  \hspace{1cm} (5)

where $g_p$ is defined as the $p$-th denoising group. Each denoising group contains $M$ queries where $M$ is the number of GT objects in the image. So we have

$$g_p = \{q^p_0, q^p_1, \ldots, q^p_{M-1}\}$$  \hspace{1cm} (6)

where $q^p_m = \delta(t_m)$.

The purpose of the attention mask is to prevent information leakage. There are two types of potential information leakage. One is that the matching part may see the noised GT objects and easily predict GT objects. The other is that one noised version of a GT object may see another version. Therefore, our attention mask is to make sure the matching part cannot see the denoising part and the denoising groups cannot see each other as shown in Fig. 4.

We use $A = [a_{ij}]_{W \times W}$ to denote the attention mask where $W = P \times M + N$. $P$ and $M$ are the number of groups and GT objects. $N$ is the number of queries in the matching part. We let the first $P \times M$ rows and columns to represent the denoising part and the latter to represent the matching part. $a_{ij} = 1$ means the $i$-th query cannot see the $j$-th query and $a_{ij} = 0$ otherwise. We devise the attention mask as follows

$$a_{ij} = \begin{cases} 1, & \text{if } j < P \times M \text{ and } |\frac{i}{M}| \neq |\frac{j}{M}|; \\ 1, & \text{if } j < P \times M \text{ and } i \geq P \times M; \\ 0, & \text{otherwise}. \end{cases}$$  \hspace{1cm} (7)

Note that whether the denoising part can see the matching part or not will not influence the performance, since the queries of matching part are learned queries that contain no information of the ground truth objects.

The extra computation introduced by multiple denoising groups is negligible—when 5 denoising groups are introduced, GFLOPs for training is only increased from 94.4 to 94.6 for DAB-DETR with R50 backbone and there is no computation overhead for testing.

4.5. Label Embedding

The decoder embedding is specified as label embedding in our model to support both box denoising and label denoising. Except for the 80 classes in COCO 2017 [10], we also consider an unknown class embedding which is used in the matching part to be semantically consistent with the denoising part. We also append an indicator to label embedding. The indicator is 1 if a query belongs to the denoising part and 0 otherwise.
queries, while DETR uses 100. For example, our DN-Deformable-DETR is built to be plugged into other DETR-like models to boost performance. We also show that denoising training can boost a CNN backbone, multiple Transformer encoder layers and decoder layers. We adopt several ResNet models [8] pre-trained on ImageNet as our backbones and report our results on 4 ResNet settings: ResNet-50 (R50), ResNet-101 (R101), and their 16x-resolution extensions ResNet-50-DC5 (DC5-R50) and ResNet-101-DC5 (DC5-R101). For hyperparameters, we follow DAB-DETR to use a 6-layer Transformer encoder and a 6-layer Transformer decoder and 256 as the hidden dimension. We add uniform noise on boxes and drop lr at the 40-th epoch.

### 5. Experiment

#### 5.1. Setup

**Dataset:** We show the effectiveness of DN-DETR on the challenging COCO 2017 [10] Detection task. Following the common practice, we report the standard mean average precision (AP) result on the COCO validation dataset under different IoU thresholds and object scales.

**Implementation Details:** We test the effectiveness of the denoising training on DAB-DETR, which is composed of a CNN backbone, multiple Transformer encoder layers and decoder layers. We also show that denoising training can be plugged into other DETR-like models to boost performance. For example, our DN-Deformable-DETR is built upon Deformable DETR in multi-scale setting.
Table 3. Best results for our DN-DETR and other detection models with the ResNet-50 backbone. * indicates it is the test-dev result.

<table>
<thead>
<tr>
<th>Model</th>
<th>MultiScale</th>
<th>#epochs</th>
<th>AP</th>
<th>AP$_{50}$</th>
<th>AP$_{75}$</th>
<th>AP$_S$</th>
<th>AP$_M$</th>
<th>AP$_L$</th>
<th>GFLOPs</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deformable DETR-R50 [20]</td>
<td>✓</td>
<td>50</td>
<td>43.8</td>
<td>62.6</td>
<td>47.7</td>
<td>26.4</td>
<td>47.1</td>
<td>58.0</td>
<td>173</td>
<td>40M</td>
</tr>
<tr>
<td>SMCA-R50 [6]</td>
<td>✓</td>
<td>50</td>
<td>43.7</td>
<td>63.6</td>
<td>47.2</td>
<td>24.2</td>
<td>47.0</td>
<td>60.4</td>
<td>152</td>
<td>40M</td>
</tr>
<tr>
<td>TSP-RCNN-R50 [16]</td>
<td>✓</td>
<td>96</td>
<td>45.0</td>
<td>64.5</td>
<td>49.6</td>
<td>29.7</td>
<td>47.7</td>
<td>58.0</td>
<td>188</td>
<td>–</td>
</tr>
<tr>
<td>Dynamic DETR-R50 [4]</td>
<td>✓</td>
<td>50</td>
<td>47.2</td>
<td>65.9</td>
<td>51.1</td>
<td>28.6</td>
<td>49.3</td>
<td>59.1</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>DAB-Deformable-DETR-R50</td>
<td>✓</td>
<td>50</td>
<td>46.9</td>
<td>66.0</td>
<td>50.8</td>
<td>30.1</td>
<td>50.4</td>
<td>62.5</td>
<td>195</td>
<td>48M</td>
</tr>
<tr>
<td>DN-Deformable-DETR-R50</td>
<td>✓</td>
<td>48.6</td>
<td>67.4</td>
<td>52.7</td>
<td>31.0</td>
<td>52.0</td>
<td>63.7</td>
<td>195</td>
<td>48M</td>
<td></td>
</tr>
</tbody>
</table>

5.3. 1× Setting

With denoising training, the detection task can be accelerated by a large margin. As shown in Table 2, we compare our method with both a traditional detector [15] and some DETR-like models [1, 4, 20]. Note that Dynamic DETR [4] adopts dynamic encoder, for a fair comparison, we also compare with its version without dynamic encoder.

Under the same setting with the DC5-R50 backbone, DN-DETR can outperform DAB-DETR by +3.7 AP within 12 epochs. Compared with other models, DN-Deformable-DETR achieves the best results in the 12 epoch setting. It is worth noting that our DN-Deformable-DETR achieve 44.1 AP within 12 epochs with the ResNet-101 backbone, which surpasses Faster R-CNN ResNet-101 trained for 108 epochs (9× faster).

5.4. Compared with State-of-Art Detectors

We also conduct experiments to compare our method with multi-scale models. The results is summarized in Table 3. Our proposed DN-Deformable-DETR achieves the best result 48.6 AP with the ResNet-50 backbone. To eliminate the performance improvement from formulating the queries of deformable DETR as anchor boxes, we further use a strong baseline DAB-Deformable-DETR without denoising training. The results show that we can still yield 1.7 AP absolute improvement. The performance improvement of DN-Deformable-DETR also indicates that denoising training can be integrated into other DETR-like models and improve their performance. Though it is not a fair comparison with Dynamic DETR as it includes dynamic encoder and more scales (5 scales) with FPN, we still yield +1.4 AP improvement.

We also show the convergence curve in both single-scale and multi-scale setting in Fig. 5, where we drop learning rate by 0.1 in multiple epochs in Fig. 5(b). The detailed training acceleration analysis and training efficiency is shown in Appendix 7.1 and 7.2.

5.5. Ablation Study

We conduct a series of ablation study with the ResNet-50 backbone trained for 50 epochs to verify the effectiveness each component and report the results in Table 4 and Table 5. The results in Table 4 show that each component in denoising training contributes to the performance improve-
Box Denoising | Label Denoising | Attention Mask | AP
---|---|---|---
✓ | ✓ | ✓ | 43.4
✓ | ✓ | ✓ | 43.0
✓ | ✓ | ✓ | 42.2
✓ | ✓ | ✓ | 41.8

Table 4. Ablation results for DN-DETR. All models are trained with the ResNet-50 backbone using 1 denoising group under the same default settings.

Limitations and Future Work: In this work, the added noises are simply sampled from uniform distribution. We have not explored more complex noising schemes and leave these for future work. Reconstructing noised data achieves great success in un-supervised learning. This work is an initial step to apply it into object detection. In the future, we will explore how to pretrain detectors on weakly labeled data with unsupervised learning techniques or explore other unsupervised learning methods such as contrastive learning.
References


