DETRs with Hybrid Matching

Ding Jia†  Yuhui Yuan‡§  Haodi He†  Xiaopei Wu
Haojun Yu†  Weihong Lin‡  Lei Sun‡  Chao Zhang†  Han Hu‡

1Peking University  2Stanford University  3Zhejiang University  4Microsoft Research Asia

Abstract

One-to-one set matching is a key design for DETR to establish its end-to-end capability, so that object detection does not require a hand-crafted NMS (non-maximum suppression) to remove duplicate detections. This end-to-end signature is important for the versatility of DETR, and it has been generalized to broader vision tasks. However, we note that there are few queries assigned as positive samples and the one-to-one set matching significantly reduces the training efficacy of positive samples. We propose a simple yet effective method based on a hybrid matching scheme that combines the original one-to-one matching branch with an auxiliary one-to-many matching branch during training. Our hybrid strategy has been shown to significantly improve accuracy. In inference, only the original one-to-one match branch is used, thus maintaining the end-to-end merit and the same inference efficiency of DETR. The method is named H-DETR, and it shows that a wide range of representative DETR methods can be consistently improved across a wide range of visual tasks, including Deformable-DETR, PETRv2, PETR, and TransTrack, among others. Code is available at: https://github.com/HDETR.

1. Introduction

Since the success of pioneering work DETection TRansformer (DETR) [5] on object detection tasks, DETR-based approaches have achieved significant progress on various fundamental vision recognition tasks such as object detection [50, 58, 82, 89], instance segmentation [13, 14, 25, 78], panoptic segmentation [9, 31, 64, 75, 79], referring expression segmentation [69, 74], video instance segmentation [8, 65, 70], pose estimation [26, 59, 60], multi-object tracking [7, 49, 61], monocular depth estimation [18, 29], text detection & layout analysis [46, 55, 56, 85], line segment detection [71], 3D object detection based on point clouds or multi-view images [1, 30, 52, 67], visual question answering [22, 47], and so on.

Many follow-up efforts have improved DETR from various aspects, including redesigning more advanced transformer encoder [15, 89] or transformer decoder architectures [4, 16, 50, 81, 89] or query formulations [24, 37, 68, 82]. Different from most of these previous efforts, we focus on the training efficacy issues caused by one-to-one matching, which only assigns one query to each ground truth. For example, Deformable-DETR typically only selects less than 30 queries from a pool of 300 queries to match with the ground truth for each image, as nearly 99% of the COCO images consist of less than 30 bounding boxes annotations, while the remaining more than 270 queries will be assigned as ∅ and are supervised with only classification loss, thus suffering from very limited localization supervision.

To overcome the drawbacks of one-to-one matching and unleash the benefits of exploring more positive queries, we present a very simple yet effective hybrid matching scheme, which introduces an additional one-to-many matching branch that assigns multiple queries to each positive sample. In inference, we only use the original one-to-one decoder branch supervised with the one-to-one matching loss. We find that this simple approach can substantially improve the training efficacy, especially regarding the fitting of positive queries. Since only the original one-to-one matching branch is used in inference, the merits of the original DETR framework are almost all maintained, for example, avoiding NMS. Our approach also has no additional computation overhead compared to the original version.

We dub the hybrid matching approach as H-DETR, and extensively verify its effectiveness using a variety of vision tasks that adopt DETR methods or the variants, as well as different model sizes ranging from ResNet-50/Swin-T to Swin-L. The visual tasks and the corresponding DETR-based approaches include Deformable-DETR [89] for image object detection, PETRv2 [40] for 3D object detection from multi-view images, PETR [59] for multi-person pose estimation, and TransTrack [61] for multi-object tracking. The H-DETR achieves consistent gains over all of them, as

†Core contribution.
‡Corresponding author.
shown in Figure 1. Specifically, our approach can improve the Deformable DETR framework (R50) on COCO object detection by +1.7% mAP (48.7% v.s. 47.0%), the PETR framework (R50) on COCO pose estimation by +1.6% mAP (70.9% v.s. 69.3%). In particular, we achieve 59.4% mAP on COCO object detection, which is the highest accuracy on COCO object detection among DETR-based methods that use the Swin-L model. We achieve 52.38% on nuScenes val, which is +1.7% higher than a very recent state-of-the-art approach of PETRv2.

2. Related work

DETR for object detection. With the pioneering work DETR [5] introducing transformers [63] to 2D object detection, more and more follow-up works [11, 15, 50, 68] have built various advanced extensions based on DETR because it removes the need for many hand-designed components like non-maximum suppression [53] or initial anchor boxes generation [17, 35, 38, 57]. Deformable-DETR [89] introduced the multi-scale deformable self/cross-attention scheme, which attends to only a small set of key sampling points around a reference and achieves better performance than DETR (especially on small objects). DAB-DETR [37] further verified that a different novel query formulation can also improve the performance. The follow-up DINO-DETR [24, 82] has established the new SOTA results on object detection tasks and demonstrated the advantages of DETR design by introducing a novel query denoising scheme. Different from these works, we focus on the matching mechanism design of DETR and propose a very simple strategy to improve the training efficacy while still avoiding NMS, which also differentiates our efforts from the very recent DE-DETR [66] that requires NMS.

DETR for other vision tasks. Inspired by the great success of DETR in object detection, many recent efforts have constructed various DETR-based approaches for segmentation tasks [8, 9, 25, 31, 76, 77] that aim for more accurate pixel localization and recognition. 3D object detection tasks [1, 20, 21, 30, 34, 39, 40, 44, 45, 52, 67] based on point-cloud or multi-view-images that target to identify and localize objects in 3D scenes, pose estimation tasks [2, 26, 28, 48, 59, 60, 83] with the objective of localizing the key points of the presented persons in a given image, object tracking tasks [49, 61, 72, 80, 86] that aim at locating the objects across both spatial and temporal positions within a given video without any prior knowledge about the appearance, and so on. To verify the generalization ability of our approach, we first construct a baseline approach, i.e., Mask-Deformable-DETR, for segmentation tasks, and then combine our hybrid matching scheme with Mask-Deformable-DETR to deliver strong panoptic segmentation results on the COCO benchmark. For 3D object detection, pose estimation, and object tracking, we directly choose the recent approaches including PETRv2 [39, 40], PETR [59], TransTrack [61] as our baseline and verify that our hybrid matching consistently improves their performance.

Label assignment. We can categorize the existing label assignment approaches, following the previous work [65, 88], into two different paths: (i) one-to-many label assignment, i.e., assigning multiple predictions as positive samples for each ground-truth box [57, 62, 84], and (ii) one-to-one label assignment, i.e., assigning only one prediction as a positive sample for each ground-truth box. POTO [65] propose to assign the anchor with either the maximum IoU or closest to the object center as the positive sample, which is modified from the strategies of RetinaNet [35] or FCOS [62]. DETR [5] and its followups [4, 24, 37, 50, 68, 89] apply the Hungarian matching to compute one-to-one positive assignments based on the global minimum matching cost values between all predictions and the ground-truth boxes. Different from the most related work POTO [65] that only uses one-to-many assignment, based on ATSS [84], to help the classification branch, our approach chooses Hungarian matching to perform both one-to-one matching and one-to-many matching following DETR and generalizes to various DETR-based approaches across vision tasks.

Relationship to DN-DETR and DINO-DETR: Our approach is also related to recent approaches that introduce noisy augmentations of ground-truth objects as auxiliary queries,
i.e., DN-DETR [24] and DINO-DETR [82]. Similar to our approach, they also introduce additional auxiliary queries. However, the aims of these approaches and ours are different: while DN-DETR and DINO-DETR mainly aim to address the instability issue of Hungarian assignment, we mainly address the insufficient training problem of positive samples in one-to-one matching.

The different aims may have led to their different designs: DN-DETR and DINO-DETR involve a noising scheme and take manual assignment between the noisy queries and ground-truth objects, while our approach is much simpler which uses an end-to-end assignment strategy to match the auxiliary query set and ground-truth set using the Hungarian method (each ground truth is repeated multiple times). The end-to-end manner also allows our approach to be more general than the methods based on denoising query: while DN-DETR/DINO-DETR needs to tune or redesign its noising scheme and query forms, our approach can be easily extended various DETR variants for different vision problems with almost no additional tuning.

3. Our Approach

3.1. Preliminary

General DETR pipeline. Given an input image $I$, DETR first applies the backbone and the transformer encoder to extract a sequence of enhanced pixel embeddings $X = \{ x_0, x_1, \cdots, x_N \}$. Second, DETR sends the enhanced pixel embeddings and a default group of object query embeddings $Q = \{ q_0, q_1, \cdots, q_n \}$ into the transformer decoder. Third, DETR applies the task-specific prediction heads on the updated object query embeddings after each transformer decoder layer to generate a set of predictions $P = \{ p_0, p_1, \cdots, p_n \}$ independently. Last, DETR performs one-to-one bipartite matching between the predictions and the ground-truth bounding boxes and labels $G = \{ g_0, g_1, \cdots, g_m \}$. Specifically, DETR associates each ground truth with the prediction that has the minimal matching cost and applies the corresponding supervision accordingly. Figure 2 also illustrates the overall pipeline of the DETR approach. The follow-up works have reformulated the object query to various variants for different visual recognition tasks such as mask query [9, 31], pose query [59], track query [49, 61], bins query [32], and so on. We use "query" in the following discussions for simplicity and consistency.

General Deformable-DETR pipeline. The Deformable-DETR improves the pipeline of DETR from the following three main aspects: (i) replace the original multi-head self-attention or cross-attention with a multi-scale deformable self-attention and multi-scale deformable cross-attention scheme; (ii) replace the original independent layer-wise prediction scheme with iterative refinement prediction scheme; (iii) replace the original image content irrelevantly query with a dynamic query generated by the output from the transformer encoder. Besides, Deformable-DETR also performs one-to-one bipartite matching following the DETR. Readers could refer to [89] for more details.

3.2. Hybrid Matching

The key idea of our hybrid matching approach is to combine the advantages of one-to-one matching scheme with those of the one-to-many matching scheme, where the one-to-one matching is necessary for removing NMS and the one-to-many matching enriches the number of queries that are matched with ground truth for higher training efficacy. We first illustrate detailed implementations of the hybrid branch scheme, and then briefly introduce the implementations of another two simple variants, including the hybrid epoch scheme and hybrid layer scheme. We summarize the pipelines of these hybrid matching schemes in Figure 3.

3.2.1 Hybrid Branch Scheme

We maintain two groups of queries $Q = \{ q_1, q_2, \cdots, q_n \}$ and $Q = \{ \hat{q}_1, \hat{q}_2, \cdots, \hat{q}_N \}$, where we apply one-to-one matching or one-to-many matching on the predictions based on $Q$ or $Q$ respectively.

One-to-one matching branch. We process the first group of queries $Q$ with $L$ transformer decoder layers and perform predictions on the output of each decoder layer respectively. Then, we perform the bipartite matching between the $\{ \text{predictions, ground-truth} \}$ pair over each layer, e.g., estimating $\mathcal{L}_{\text{match}}(P^l, G)$, and compute the loss as follows:

$$\mathcal{L}_{\text{one2one}} = \sum_{l=1}^{L} \mathcal{L}_{\text{Hungarian}}(P^l, G),$$  \hspace{1cm} (1)$$

where $P^l$ represents the predictions outputted by the $l$-th transformer decoder layer. We choose $\mathcal{L}_{\text{match}}(\cdot)$ and $\mathcal{L}_{\text{Hungarian}}(\cdot)$ following DETR [5] and Deformable-DETR [89], which consist of a classification loss, a $\mathcal{L}_1$ regression loss, and a GIoU loss.
Figure 3. Illustrating the pipeline of our Hybrid Matching scheme. We use the colored regions of the same color to mark their parameters are shared. We use $\rho$ to represent the percentage of training epochs. We have $L = L_1 + L_2$ in hybrid layer scheme.

**One-to-many matching branch.** Then, we process the second group of queries $Q$ with the same $L$ transformer decoder layers and get $L$ groups of predictions. In order to perform one-to-many matching, we simply repeat the ground truth for $K$ times and get an augmented target $G = \{G^1, G^2, \ldots, G^K\}$, where $G^1 = G^2 = \cdots = G^K = G$. We also perform the bipartite matching between the predictions, augmented ground truth pair over each layer, e.g., estimating $L_{\text{match}}(\hat{P}^l, G)$, and compute the corresponding loss as follows:

$$L_{\text{one2many}} = \sum_{l=1}^{L} L_{\text{Hungarian}}(\hat{P}^l, G),$$  

where $\hat{P}^l$ represents the predictions output by the $l$-th transformer decoder layer.

In summary, we use the combination of the above two losses, i.e., $\lambda L_{\text{one2many}} + L_{\text{one2one}}$, through the whole training process. To accelerate the training speed and process both $Q$ or $Q$ in parallel, we further apply a masked multi-head self-attention to avoid their interactions. Therefore, we do not observe significant extra training costs in experiments. We provide detailed comparison results in the following Table 6 within the experiment section. Last, we only keep the one-to-one matching branch, i.e., $Q$, during evaluation. We present the overall pipeline on the left of Figure 3.

### 3.2.2 More Variants of Hybrid Matching

**Hybrid epoch scheme.** Different from the hybrid branch scheme, we only maintain a single group of queries $Q = \{q_1, q_2, \cdots, q_M\}$, where we apply both one-to-one matching and one-to-many matching on the predictions based on $Q$ during different training epochs. We illustrate more details as follows.

- **One-to-many matching training epochs:** During the first $\rho$ training epochs, we perform one-to-many matching to process $Q$ with $L$ transformer decoder layers and get $L$ groups of predictions. We also get the augmented ground truth $\hat{G} = \{G^1, G^2, \cdots, G^K\}$ following the similar manner adopted by the one-to-many matching branch. Then we perform the bipartite matching between $L_{\text{match}}(\hat{P}^l, \hat{G})$ and compute the loss as follows:

$$L_{\text{one2many}} = \sum_{l=1}^{L} L_{\text{Hungarian}}(\hat{P}^l, \hat{G}).$$

- **One-to-one matching training epochs:** We change one-to-many matching to one-to-one matching for the remaining $(1 - \rho)$ training epochs. The only difference is that we match the predictions with the original ground truth and illustrate the formulation as follows:

$$L_{\text{one2one}} = \sum_{l=1}^{L} L_{\text{Hungarian}}(P^l, G).$$

Last, we directly apply $\hat{Q}$ during evaluations without using NMS. In summary, we apply only $L_{\text{one2many}}$ or $L_{\text{one2one}}$ in the first $\rho$ training epochs or last $(1 - \rho)$ training epochs respectively. We also illustrate the overall pipeline of the hybrid epoch scheme in the middle of Figure 3.

**Hybrid layer scheme.** Similar to the hybrid epoch scheme, we also maintain a single group of queries $Q = \{q_1, q_2, \cdots, q_N\}$. Instead of performing different matching strategies across different training epochs, we apply one-to-many matching on the prediction output by the first $L_1$ transformer decoder layers and one-to-one matching on the prediction output by the remaining $L_2$ transformer decoder layers.

- **One-to-many matching decoder layers:** We choose to apply a one-to-many matching scheme for the first $L_1$ transformer decoder layers, where we supervise the predictions, output by each one of the first $L_1$ layers, with the augmented ground truth $G = \{G^1, G^2, \cdots, G^K\}$ following:

$$L_{\text{one2many}} = \sum_{l=1}^{L_1} L_{\text{Hungarian}}(P^l, G),$$

where we also need to perform the bipartite matching between $L_{\text{match}}(\hat{P}, \hat{G})$ before computing the above loss.

- **One-to-one matching decoder layers:** For the following $L_2$ transformer decoder layers, we perform a one-to-one
matching scheme on their predictions as follows:

\[ L_{\text{one2many}} = \sum_{l=1}^{L_1+L_2} L_{\text{Hungarian}}(P^l, G). \] (6)

In summary, we apply the combination of both \( L_{\text{one2many}} \) and \( L_{\text{one2one}} \) through the whole training procedure. The right of Figure 3 presents the overall pipeline.

4. Experiment

4.1. Improving DETR-based Approaches

2D object detection results. Table 1 reports the comparison results on the COCO object detection \( \text{val} \) set. Our \( \mathcal{H} \)-Deformable-DETR achieves consistent gains over the baseline with backbones of different scales (including ResNet-50, Swin-T and Swin-L) trained under 12 epochs or 36 epochs. For example, when choosing Swin-T under 12 and 36 training epochs, our \( \mathcal{H} \)-Deformable-DETR improves Deformable-DETR from 51.8\% to 53.2\% and 49.3\% to 50.6\%, respectively.

Besides, we report the comparison results on LVIS object detection in Table 2. Our approach also achieves consistent gains over the baseline Deformable-DETR across various backbones, e.g., with Swin-L as the backbone, \( \mathcal{H} \)-Deformable-DETR improves the AP score by +0.9\%.

3D object detection results. We choose the very recent representative DETR-based approach, i.e., PETRv2 [40], to verify the generalization ability of our approach for 3D detection based on multi-view images. Table 3 summarizes the detailed comparison results. We can see that our \( \mathcal{H} \)-PETRv2 significantly improves the NDS scores of baseline PETRv2 from 50.68\% to 52.38\% on nuScenes \( \text{val} \), thus showing that our hybrid matching improves the localization accuracy of 3D object detection predictions. We observe the GPU memory consumption increases from 7235M to 11175M, where 78% of the increased GPU memory locates at the cross-attention. A preliminary optimization by sequential self/cross attention [41] has reduced the memory consumption from 11175M to 8733M while maintaining the performance.

Multi-person pose estimation results. We extend our hybrid matching strategy to the very recent PETR (Pose Estimation with TRansformers) [59] and summarize the detailed results in Table 4. We can see that our approach achieves consistent gains over the baselines. For example, with Swin-L as the backbone, our \( \mathcal{H} \)-PETR improves the AP score of PETR from 73.3\% to 74.9\% on the COCO \( \text{val} \) under even 100 training epochs.

Multi-object tracking results. We apply our hybrid matching scheme to a powerful multi-object tracking approach, e.g., TransTrack [61]. Table 5 summarizes the detailed comparison results. We find the results on MOT17 suffer from relatively large variance, thus we report the mean performance with \( \sim 3 \) runs. Accordingly, our approach also shows strong potential on the multi-object tracking tasks, and our \( \mathcal{H} \)-TransTrack improves the MOTA score of TransTrack from 67.1\% to 68.7\% on MOT17 \( \text{val} \), where we also observe that the gains are related to much lower false negative rates (FN).

Panoptic segmentation results. We also report the detailed COCO panoptic segmentation results in the supplementary to verify the generalization ability of our approach.

4.2. Ablation Study

Comparing different hybrid matching schemes. We first choose a two-stage Deformable-DETR (with increased FFN dimension) as the baseline and compare the results based on our three different hybrid matching approaches. To ensure fairness, we choose their settings as follows:

- Baseline settings: We use 300 or 1800 queries and apply the conventional one-to-one matching following the original Deformable-DETR [89]. To ensure more fair comparisons with our hybrid branch scheme, we further report the baseline results under 1.3\( \times \) and 1.15\( \times \) training epochs to

<table>
<thead>
<tr>
<th>method</th>
<th>backbone</th>
<th>#epochs</th>
<th>AP</th>
<th>AP_S</th>
<th>AP_M</th>
<th>AP_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deformable-DETR</td>
<td>R50</td>
<td>12</td>
<td>47.0</td>
<td>29.1</td>
<td>50.0</td>
<td>61.6</td>
</tr>
<tr>
<td>( \mathcal{H} )-Deformable-DETR</td>
<td>R50</td>
<td>12</td>
<td>48.7\textsuperscript{1.7}</td>
<td>31.2</td>
<td>51.5</td>
<td>63.5</td>
</tr>
<tr>
<td>Deformable-DETR</td>
<td>Swin-T</td>
<td>12</td>
<td>49.3</td>
<td>31.6</td>
<td>52.4</td>
<td>64.6</td>
</tr>
<tr>
<td>( \mathcal{H} )-Deformable-DETR</td>
<td>Swin-T</td>
<td>12</td>
<td>50.6\textsuperscript{1.3}</td>
<td>33.4</td>
<td>53.7</td>
<td>65.9</td>
</tr>
<tr>
<td>Deformable-DETR</td>
<td>Swin-L</td>
<td>12</td>
<td>54.5</td>
<td>37.0</td>
<td>58.6</td>
<td>71.0</td>
</tr>
<tr>
<td>( \mathcal{H} )-Deformable-DETR</td>
<td>Swin-L</td>
<td>12</td>
<td>55.9\textsuperscript{1.4}</td>
<td>39.1</td>
<td>59.9</td>
<td>72.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>method</th>
<th>backbone</th>
<th>#epochs</th>
<th>AP</th>
<th>AP_S</th>
<th>AP_M</th>
<th>AP_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deformable-DETR</td>
<td>R50</td>
<td>36</td>
<td>49.0</td>
<td>32.6</td>
<td>52.3</td>
<td>63.3</td>
</tr>
<tr>
<td>( \mathcal{H} )-Deformable-DETR</td>
<td>R50</td>
<td>36</td>
<td>50.0\textsuperscript{1.0}</td>
<td>32.9</td>
<td>52.7</td>
<td>65.3</td>
</tr>
<tr>
<td>Deformable-DETR</td>
<td>Swin-T</td>
<td>36</td>
<td>51.8</td>
<td>34.8</td>
<td>55.1</td>
<td>67.8</td>
</tr>
<tr>
<td>( \mathcal{H} )-Deformable-DETR</td>
<td>Swin-T</td>
<td>36</td>
<td>53.2\textsuperscript{1.4}</td>
<td>35.9</td>
<td>56.4</td>
<td>68.2</td>
</tr>
<tr>
<td>Deformable-DETR</td>
<td>Swin-L</td>
<td>36</td>
<td>56.3</td>
<td>39.2</td>
<td>60.4</td>
<td>71.8</td>
</tr>
<tr>
<td>( \mathcal{H} )-Deformable-DETR</td>
<td>Swin-L</td>
<td>36</td>
<td>57.1\textsuperscript{0.8}</td>
<td>39.7</td>
<td>61.4</td>
<td>73.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>method</th>
<th>backbone</th>
<th>#epochs</th>
<th>AP</th>
<th>AP_S</th>
<th>AP_M</th>
<th>AP_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deformable-DETR</td>
<td>R50</td>
<td>24</td>
<td>32.2</td>
<td>23.2</td>
<td>41.6</td>
<td>49.3</td>
</tr>
<tr>
<td>( \mathcal{H} )-Deformable-DETR</td>
<td>R50</td>
<td>24</td>
<td>33.5\textsuperscript{1.3}</td>
<td>24.1</td>
<td>42.4</td>
<td>50.2</td>
</tr>
<tr>
<td>Deformable-DETR</td>
<td>Swin-L</td>
<td>48</td>
<td>47.0</td>
<td>35.9</td>
<td>57.8</td>
<td>66.9</td>
</tr>
<tr>
<td>( \mathcal{H} )-Deformable-DETR</td>
<td>Swin-L</td>
<td>48</td>
<td>47.9\textsuperscript{0.9}</td>
<td>36.3</td>
<td>58.6</td>
<td>67.9</td>
</tr>
</tbody>
</table>
Table 3. Multi-view 3D detection results on nuScenes.

<table>
<thead>
<tr>
<th>method</th>
<th>backbone</th>
<th>#epochs</th>
<th>mAP</th>
<th>NDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PETRv2 [40]</td>
<td>VoVNet-99</td>
<td>24</td>
<td>41.04</td>
<td>50.25</td>
</tr>
<tr>
<td>PETRv2 (Our repro.)</td>
<td>VoVNet-99</td>
<td>24</td>
<td>40.41</td>
<td>49.69</td>
</tr>
<tr>
<td>H-PETRv2</td>
<td>VoVNet-99</td>
<td>24</td>
<td>41.93$^{\pm.52}$</td>
<td>51.23</td>
</tr>
<tr>
<td>PETRv2 (Our repro.)</td>
<td>VoVNet-99</td>
<td>36</td>
<td>41.07</td>
<td>50.68</td>
</tr>
<tr>
<td>H-PETRv2</td>
<td>VoVNet-99</td>
<td>36</td>
<td>42.59$^{\pm.52}$</td>
<td>52.38</td>
</tr>
</tbody>
</table>

Table 4. Multi-person pose estimation results on COCO.

| method | backbone | #epochs | AP  | AP$$_M$$ | AP$_L$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PETR [59]</td>
<td>R50</td>
<td>100</td>
<td>68.8</td>
<td>62.7</td>
<td>77.7</td>
</tr>
<tr>
<td>PETR (Our repro.)</td>
<td>R50</td>
<td>100</td>
<td>69.3</td>
<td>63.3</td>
<td>78.4</td>
</tr>
<tr>
<td>H-PETR</td>
<td>R50</td>
<td>100</td>
<td>70.9$^{+1.6}$</td>
<td>64.4</td>
<td>80.3</td>
</tr>
<tr>
<td>PETR [59]</td>
<td>R101</td>
<td>100</td>
<td>70.0</td>
<td>63.6</td>
<td>79.4</td>
</tr>
<tr>
<td>PETR (Our repro.)</td>
<td>R101</td>
<td>100</td>
<td>69.9</td>
<td>63.4</td>
<td>79.4</td>
</tr>
<tr>
<td>H-PETR</td>
<td>R101</td>
<td>100</td>
<td>71.0$^{+1.1}$</td>
<td>64.7</td>
<td>80.2</td>
</tr>
<tr>
<td>PETR [59]</td>
<td>Swin-L</td>
<td>100</td>
<td>73.1</td>
<td>67.2</td>
<td>81.7</td>
</tr>
<tr>
<td>PETR (Our repro.)</td>
<td>Swin-L</td>
<td>100</td>
<td>73.3</td>
<td>67.7</td>
<td>81.6</td>
</tr>
<tr>
<td>H-PETR</td>
<td>Swin-L</td>
<td>100</td>
<td>74.9$^{+1.6}$</td>
<td>69.3</td>
<td>83.3</td>
</tr>
</tbody>
</table>

Table 5. Multi-object tracking results on MOT.

<table>
<thead>
<tr>
<th>method</th>
<th>#epochs</th>
<th>MOTA (↑)</th>
<th>IDF1 (↑)</th>
<th>FN (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOT17 val</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TransTrack [61]</td>
<td>20</td>
<td>67.1</td>
<td>70.3</td>
<td>15820</td>
</tr>
<tr>
<td>TransTrack (Our repro.)</td>
<td>20</td>
<td>67.1</td>
<td>68.1</td>
<td>15680</td>
</tr>
<tr>
<td>H-TransTrack</td>
<td>20</td>
<td>68.7$^{+1.6}$</td>
<td>68.3</td>
<td>13657</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>method</th>
<th>#epochs</th>
<th>MOT17 test</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSTrack</td>
<td>20</td>
<td>74.5</td>
<td>63.9</td>
<td>112137</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H-TRANSTrack</td>
<td>20</td>
<td>74.6</td>
<td>63.2</td>
<td>111105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H-TRANSTrack</td>
<td>20</td>
<td>75.7$^{+1.1}$</td>
<td>64.4</td>
<td>91155</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ensure even more total training time than our hybrid branch scheme. We empirically find that the increased time mainly comes from two aspects including more queries (1800 vs. 300) and the extra Hungarian matching operation and loss computation of one-to-many matching branch. By simply merging the matching cost computation and loss computation of one-to-one matching branch and one-to-many matching branch, our hybrid-branch scheme only bring around $+7\%$ extra training time compared to the baseline with 1800 queries. The extra time could be further decreased by implementing the Hungarian matching on GPU instead of CPU, which is not the focus of this work.

- Hybrid branch settings: We use 300 queries ($n=300$) and 1500 queries ($T=1500$) for one-to-one matching and one-to-many matching branch, respectively. We set $K$ as 6 for the augmented ground truth $\hat{G}$. Thus we will generate $6\times$ additional positive queries with the one-to-many matching branch, e.g., we have $(1+6)\times12\times6$ under 12 training epochs if the benchmark is the number of positive queries within one training epoch for each decoder layer.

- Hybrid epoch settings: We use 1800 queries through the whole training process, i.e., $M=1800$. To generate $6\times$ additional positive queries than the baseline, we set $K=10$ and $\rho=\frac{2}{3}$. Therefore, we have $10\times\frac{2}{3}\times12\times6+\frac{1}{3}\times12\times6=(1+6)\times12\times6$ holds under 12 training epochs.

- Hybrid layer settings: We also use 1800 queries through the whole training process, i.e., $N=1800$. To generate $6\times$ additional positive queries than the baseline, we set $K=10$, $L_1=4$, and $L_2=2$ considering that $10\times12\times4+12\times2=(1+6)\times12\times6$ holds.

Table 6 summarizes the detailed comparison results. Accordingly, we find that hybrid matching approaches consistently outperform the baseline under different training epochs (including $1.3\times$ and $1.15\times$ training epochs). We also report the GPU memory of hybrid-branch/hybrid-layer/hybrid epoch (for ResNet50): 7728M/7722M/7711M. We can see that the hybrid branch scheme is the best choice if we consider the trade-off of training time, inference
Table 7. GFLOPs/training time/GPU memory vs. # query based on our hybrid branch scheme. The numbers within () are based on Swin-L.

<table>
<thead>
<tr>
<th>Backbone method</th>
<th># query [hyper-parameters]</th>
<th>GFLOPs</th>
<th>training time (min)</th>
<th>GPU memory (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>300 = [300,7=0,K=60]</td>
<td>368.19</td>
<td>65 (202)</td>
<td>5480 (8955)</td>
</tr>
<tr>
<td>ResNet50 (Swin-L)</td>
<td>600 = [300,7=300,K=60]</td>
<td>271.24</td>
<td>71 (205)</td>
<td>5719 (9150)</td>
</tr>
<tr>
<td>Ours</td>
<td>900 = [300,7=600,K=60]</td>
<td>274.03</td>
<td>72 (206)</td>
<td>6045 (9510)</td>
</tr>
<tr>
<td></td>
<td>1200 = [300,7=900,K=60]</td>
<td>276.82</td>
<td>75 (210)</td>
<td>6269 (10086)</td>
</tr>
<tr>
<td></td>
<td>1500 = [300,7=1200,K=60]</td>
<td>279.69</td>
<td>78 (215)</td>
<td>7071 (10558)</td>
</tr>
<tr>
<td></td>
<td>2000 = [300,7=1500,K=60]</td>
<td>282.39</td>
<td>80 (217)</td>
<td>7378 (11293)</td>
</tr>
</tbody>
</table>

GFLOPsFPS, and accuracy. Therefore, we choose the hybrid branch scheme (for its faster training and inference) by default if not specified in the following ablation experiments.

About computation/training time/GPU memory cost. Although $H$-Deformable-DETR uses $6 \times$ number of queries, it has small training overhead on the original Deformable-DETR framework: 1.6%/5.4% on GFLOPs and 6.4%/23.1% on training times for Swin-L/ResNet50, respectively (see Table 7). This is because only a portion of the decoder computation is affected by the number of queries, which takes about 0.3%/1.1% of the entire network FLOPs for Swin-L/ResNet50. Also note that the more increases in training time is caused by the Hungarian matching (implemented on CPU) and the cost/loss computation steps, which are inessential, and we leave its optimization as our future work.

Most of the GPU memory consumption overhead by our method comes from the naive self/cross-attention modules. The large memory complexity is not intrinsic, and it can be significantly reduced by advanced implementation, e.g., Flash Attention [12] can decrease the memory to linear complexity, and the sequential attention technique in SwinV2 can also greatly reduce the memory consumption for both self/cross attention. Both implementations are equivalent to the original one, thus not compromising accuracy. With these optimized implementation, the memory consumption increases by more queries are mild.

Effect of each component based on Deformable-DETR. We choose two-stage Deformable-DETR as our baseline and report the detailed improvements of each component in Table 8. Notably, we do not observe the obvious benefits of applying drop-out rate as zero, mixed query selection, and looking forward twice in other tasks. Therefore, we only apply this combination of tricks on 2D object detection tasks by default if not specified. We further analyze the detailed improvements of our approach based on the metrics of optimal localization recall precision in Figure 4, where the lower the better. In summary, we can see that our approach mainly improves the performance from two aspects, including more accurate localization and fewer false negatives.

1We implement both mixed query selection and looking forward twice following DINO [82].

Table 8. Comparison results based on two-stage Deformable-DETR on COCO 2017 val under 12 training epochs. 2×FFN: increase FFN dimension from 1,024 to 2,048. DP0: setting the drop out rate within transformer as 0. MQS: mixed query selection. LFT: look forward twice. HM: our hybrid matching.

| $K$ | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | AP   | 47.0 | 46.4 | 46.7 | 48.1 | 48.4 | 48.3 | 48.6 | 48.5 | 48.6 |
|-----|----|----|----|----|----|----|----|----|----|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|---
Table 11. System-level comparisons with the leading single-scale evaluation results on COCO val.

<table>
<thead>
<tr>
<th>method</th>
<th>framework</th>
<th>backbone</th>
<th>input size</th>
<th>#epochs</th>
<th>AP</th>
<th>AP50</th>
<th>AP75</th>
<th>APs</th>
<th>APM</th>
<th>APL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swin [42]</td>
<td>HTC</td>
<td>Swin-L</td>
<td>1600 × 1200</td>
<td>36</td>
<td>57.1</td>
<td>75.6</td>
<td>62.5</td>
<td>42.4</td>
<td>60.7</td>
<td>71.1</td>
</tr>
<tr>
<td>CBNetV2 [33]</td>
<td>HTC</td>
<td>2× Swin-L</td>
<td>1600 × 1400</td>
<td>12</td>
<td>59.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ConvNeXt [43]</td>
<td>Cascade Mask R-CNN</td>
<td>ConvNeXt-XL</td>
<td>1333 × 800</td>
<td>36</td>
<td>55.2</td>
<td>74.2</td>
<td>59.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MViTv2 [27]</td>
<td>Cascade Mask R-CNN</td>
<td>MViTv2-L</td>
<td>1333 × 800</td>
<td>50</td>
<td>55.8</td>
<td>74.3</td>
<td>64.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MOAT [73]</td>
<td>Cascade Mask R-CNN</td>
<td>MOAT-3</td>
<td>1344 × 1344</td>
<td>36</td>
<td>59.2</td>
<td>77.8</td>
<td>60.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Group-DETR [6]</td>
<td>DETR</td>
<td>Swin-L</td>
<td>1333 × 800</td>
<td>36</td>
<td>58.4</td>
<td>-</td>
<td>41.0</td>
<td>62.5</td>
<td>73.9</td>
<td></td>
</tr>
<tr>
<td>DINO-DETR [82]</td>
<td>DETR</td>
<td>Swin-L</td>
<td>1333 × 800</td>
<td>36</td>
<td>58.5</td>
<td>77.0</td>
<td>64.1</td>
<td>41.5</td>
<td>62.3</td>
<td>74.0</td>
</tr>
<tr>
<td>(H)-Deformable-DETR</td>
<td>DETR</td>
<td>Swin-L</td>
<td>1333 × 800</td>
<td>36</td>
<td>59.4</td>
<td>77.8</td>
<td>65.4</td>
<td>43.1</td>
<td>63.1</td>
<td>74.2</td>
</tr>
</tbody>
</table>

Table 12. Influence of sharing parameters on COCO 2017 val. under 12 training epochs.

<table>
<thead>
<tr>
<th>trans. encoder</th>
<th>trans. decoder</th>
<th>box head</th>
<th>cls head</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>48.7</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>48.5</td>
</tr>
<tr>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>48.3</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>47.3</td>
</tr>
</tbody>
</table>

Table 13. Comparison to only using one-to-many matching.

<table>
<thead>
<tr>
<th>method</th>
<th>NMS</th>
<th>train time (average)</th>
<th>inference FPS</th>
<th>#epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only one-to-many matching</td>
<td>✓</td>
<td>95min</td>
<td>5.3</td>
<td>49.4</td>
</tr>
<tr>
<td>Ours (one-to-one branch)</td>
<td>✗</td>
<td>6.7</td>
<td>48.7</td>
<td>49.9</td>
</tr>
<tr>
<td>Ours (one-to-one branch)</td>
<td>✓</td>
<td>5.6</td>
<td>48.7</td>
<td>50.0</td>
</tr>
<tr>
<td>Ours (one-to-many branch)</td>
<td>✗</td>
<td>6.5</td>
<td>13.5</td>
<td>13.1</td>
</tr>
<tr>
<td>Ours (one-to-many branch)</td>
<td>✓</td>
<td>5.4</td>
<td>18.6</td>
<td>49.8</td>
</tr>
</tbody>
</table>

or bounding box heads does not hurt the performance, (ii) further using independent transformer decoder layers slightly drops, and (iii) further using independent transformer encoder layers results in significant performance drops considering the baseline performance is 47.0%. Accordingly, we can see that the better optimization of transformer encoder instead of decoder is the key to the performance improvements.

To verify whether this observation still holds on DINO-DETR, we also report the detailed ablation results in the supplementary to verify that most of the gains of (contrast) query denoising essentially come from the better optimization of transformer encoder. For example, using independent transformer decoders and independent box & classification heads only suffers from 0.1% drop while further using an independent transformer encoder contributes to another 0.5% drop.

Comparison to only using one-to-many matching. Table 13 compares our approach to a strong variant that applies only one-to-many matching through the whole training process and NMS during evaluation. We also ablate the predictions based only on the one-to-one matching branch or one-to-many matching branch in Table 13. Accordingly, we see that (i) the one-to-many branch evaluated with NMS achieves comparable performance with the one-to-one matching branch; (ii) only using one-to-many matching evaluated with NMS achieves slightly better performance than ours; (iii) the one-to-one matching branch within our hybrid matching scheme is the best choice when considering the training time and inference speed (FPS).

We further conduct more ablation experiments including: analyzing the training/validation loss curves of the one-to-one matching branch, the influence of loss-weight on one-to-many matching branch, effect of a careful query selection scheme, precision-recall curves, and so on.

4.3. Comparison with State-of-the-art

Table 11 reports the system-level comparisons to some representative state-of-the-art methods that use single-scale evaluation on COCO val and choose backbones of similar capacity with Swin-L. By introducing additional enhancement techniques in [82], our \(H\)-Deformable-DETR achieves 59.4% on COCO val set, which surpasses the very recent DINO-DETR method, as well as other top-performing methods.

5. Conclusion

This paper presents a very simple yet surprisingly effective hybrid matching scheme to address the low training efficiency of DETR-based approaches on multiple vision tasks. Our approach explicitly combines the advantages of a one-to-one matching scheme, i.e., avoiding NMS, and those of a one-to-many matching scheme, i.e., increasing the number of positive queries and training efficacy. We hope our initial efforts can accelerate the advancement of DETR approaches on various vision tasks.

Acknowledgement Ding Jia and Chao Zhang are supported by the National Nature Science Foundation of China under Grant 2018AAA0100300.
References


[14] Yuxin Fang, Shusheng Yang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Instances as queries. In ICCV, 2021. 1


[27] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Iitendra Malik, and Christoph Feichtenhofer. Improved multiscale vision transformers for classification and detection. arXiv:2112.01526, 2021. 8


[34] Tingting Liang, Hongwei Xie, Kaicheng Yu, Zhongyu Xia, Zhiwei Lin, Yongtao Wang, Tao Tang, Bing Wang, and Zhi


Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Aksan. One metric to measure them all: Localization recall precision (lrp) for evaluating visual detection tasks. TPAMI, 2021.


[84] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020. 2

[85] Xiang Zhang, Yongwen Su, Subarna Tripathi, and Zhuowen Tu. Text spotting transformers. In *CVPR*, 2022. 1


Supplementary

A. Datasets

COCO [36]. The COCO object detection dataset consists of 123K images with 806K annotated bounding boxes belonging to 80 thing classes and 53 stuff classes, where the train set contains 118K images and the val set contains 5K images. We report the 2D object detection performance on the val set. The COCO pose estimation benchmark consists of more than 200K images and 250K person instances labeled with 17 keypoints, where train set consists of 57K images and 150K person instances, val set consists of 5K images, and test-dev set consists of 20K images, respectively.

LVIS [19]. This dataset consists of 100K images annotated with both object detection bounding boxes and instance segmentation masks for 1203 classes. We follow the very recent work Detic [87] to only use the bounding box supervision for training.

nuScenes [3]. This is a large-scale autonomous driving dataset consisting of 1000 driving sequences with each one about 20s long, where the multimodal dataset is collected from 6 cameras, 1 lidar, and 5 radars. We partition the 1000 driving sequences to 700, 150, and 150 sequences for train, val, and test respectively. We report both the numbers of nuScenes Detection Score (NDS) and mean Average Precision (mAP) to measure the 3D object detection performance.

ScanNetV2 [10]. This dataset consists of 1513 indoor scenes labeled with per-point instance ids, semantic categories and 3D bounding boxes for around 18 categories. We use 1201 and 312 scenes for train and val, respectively. We mainly report the mAP scores under two different IoU thresholds, i.e., 0.25 and 0.5.

MOT17 [51]. This dataset consists of 14 pedestrian tracking videos annotated with rich bounding boxes and their corresponding track ids. We use 7 videos for train and the other 7 videos for test. Following TransTrack [61], we split the second half of each train video to form a val set. We report the multi-object tracking performance on both val and test.

B. More Hyper-parameter Details

We illustrate the detailed hyper-parameter settings when applying our hybrid branch approach to different DETR-based approaches and different benchmarks in Table 14.

C. Panoptic Segmentation Results

To verify the effectiveness of our approach to the panoptic segmentation task, we first construct a simple baseline, i.e., Mask-Deformable-DETR, by adding a mask prediction head to the Deformable-DETR. Then, we apply our hybrid branch scheme to the Mask-Deformable-DETR following the $\mathcal{H}$-Deformable-DETR, thus resulting in $\mathcal{H}$-Mask-Deformable-DETR. We summarize the detailed comparison results in Table 15. Accordingly, we can see that our hybrid matching scheme consistently improves ResNet50, Swin-T, and Swin-L under various training epochs.

For example, $\mathcal{H}$-Mask-Deformable-DETR gains +0.3% over Mask-Deformable-DETR equipped with Swin-L when trained for 50 epochs. We guess the relatively limited gains when compared to object detection tasks, are due to that the transformer encoder receives much more dense and informative training signals from the extra introduced mask branch, which consists of the dense interactions between the query embeddings and the high-resolution feature maps output by the transformer encoder. We also have verified that better training of the transformer encoder is the key factor to the performance improvements.

D. More Ablation Results

- First, we illustrate the curves of training losses and validation losses based on Deformable-DETR in Figure 5. We
Figure 5. Illustrating the loss curves of Deformable-DETR on train and val. We can see that, with longer training epochs, e.g., from 50 epochs to 75 epochs, the training loss consistently decreases while the validation loss saturates on COCO object detection benchmark.

Figure 6. Illustrating the AP scores of Deformable-DETR on train and val under longer training epochs. We can see that, with longer training epochs, e.g., from 50 epochs to 75 epochs, the AP scores on train set consistently improves while the AP scores on val set saturates on COCO object detection benchmark.

Table 16. Influence of $\lambda$ of our approach on COCO 2017 val under 12x epochs training schedule. We set $K = 6$ and $T = 1500$.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.5</th>
<th>1</th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>47.8</td>
<td>48.0</td>
<td>48.4</td>
<td>48.7</td>
<td>48.5</td>
<td>48.3</td>
</tr>
</tbody>
</table>

Table 17. Influence of sharing parameters for DINO framework.

<table>
<thead>
<tr>
<th>trans. encoder</th>
<th>trans. decoder</th>
<th>box head</th>
<th>cls head</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>49.1</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>48.9$^{0.2}$</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>49.2$^{0.01}$</td>
</tr>
<tr>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>49.0$^{0.01}$</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>48.5$^{0.06}$</td>
</tr>
</tbody>
</table>

also report the detection performance of Deformable-DETR under various training schedules from 12x epochs to 75x epochs in Figure 6. Accordingly, we observe that (i) simply increasing the number of training epochs fails to improve the performance, and (ii) Deformable-DETR can not benefit from exploring more positive queries that are matched with ground truth during the additional training epochs, which further verifies the advantage of our hybrid matching scheme.

- Second, we study the influence of the loss weight $\lambda$ associated with the one-to-many matching loss $L_{\text{one2many}}$ in Table 16. We can see that our approach is not sensitive to the choice of $\lambda$ and we simply choose $\lambda=1$ for all experiments by default.

- Third, to investigate why hybrid matching fails to improve the performance when choosing $K = 1$. We empirically find that the bounding boxes located at the top 300 - 600 are of poor localization quality, where simply replacing the top 0 - 300 bounding box predictions with top 300 - 600 bounding box predictions during training and evaluation causes a significant performance drop (47% vs. 42.5%). To address this problem, we study the influence of a more advanced selection scheme within one-to-many matching branch: (i) use two independent classification heads on the output of the transformer encoder to perform one-to-one/one-to-many (here we use $K + 1$ groups of ground truth in one-to-many branch) matching and select top 300/300x$(K + 1)$ predictions respectively, (ii) use the remaining 300x$(K)$ predictions in the one-to-many matching branch by filtering out the duplicated ones presented in the top 300 predictions of the one-to-one matching branch. Based on the above-improved selection scheme, we achieve 47.9% when choosing $K = 1$ and observe no advantages when choosing $K = 6$ compared to the original selection strategy.

- Fourth, to verify whether this observation still holds on DINO-DETR, Table 17 reports the detailed ablation results on DINO-DETR. We can see that the gains of (contrast) query denoising also mainly comes from the better optimization of the transformer encoder instead of the decoder.
Algorithm 1 Hybrid Matching Loss Function.

```python
def naive_hybrid_loss(
    outputs_one2one, 
    outputs_one2many, 
    targets, 
    k_repeated_targets):
    loss_dict_one2one = criterion(
        outputs_one2one, targets) 
    loss_dict_one2many = criterion(
        outputs_one2many, k_repeated_targets)
    def criterion(
        prediction, 
        target):
        cost_matrix = compute_cost()
        indices = hungarian_matching(cost_matrix)
        loss(prediction, target, indices)
    def optimized_hybrid_loss(
        outputs_one2one, 
        outputs_one2many, 
        targets, 
        k_repeated_targets):
        cost_matrix_one2one, cost_matrix_one2many =
            compute_cost(
                concatenate(outputs_one2one, outputs_one2many),
                concatenate(targets, k_repeated_targets)
            ) 
        indices_one2one = hungarian_matching(
            cost_matrix_one2one)
        indices_one2many = hungarian_matching(
            cost_matrix_one2many)
        loss(
            concatenate(outputs_one2one, outputs_one2many),
            concatenate(target, k_repeated_targets),
            concatenate(indices_one2one, indices_one2many))
```

For example, using independent transformer decoders and independent box & classification heads only suffers from 0.1% drop (49.1% → 49.0%) while further using an independent transformer encoder contributes to another 0.5% drop (49.1% → 48.5%).

- Fifth, we illustrate the loss curves of our $\mathcal{K}$-Deformable-DETR and the baseline Deformable-DETR in Figure 8. To ensure fairness, we only consider the $L_{one2one}$ through the whole training procedure. Accordingly, we can see our approach achieves both lower training loss values and lower validation loss values, which shows that the additional one-to-many matching branch could ease the optimization of the one-to-one matching branch.

- Sixth, we compare the precision-recall curves of the base-line and our hybrid matching scheme in Figure 7. It can be seen that our approach mainly improves the recall rate of the baseline method, thus also ensuring the lower false negative rates.

E. Accelerate Hybrid Matching

In the original implementation, we simply perform the one-to-one/one-to-many matching and loss computation with two independent functions, thus increasing the overall training time of each epoch from 75min to 85min when compared to the baseline that uses the same total number of queries. To decrease the additional latency, we merge the computation of their cost matrices and loss functions in Algorithm 1. Based on the accelerated implementation, we decrease the original training time from 85min to 80min.

Acknowledgement

We thank Di He, Zheng Zhang, Jin-Ge Yao, Yue Cao, and Yili Wang for helpful discussions. Especially, Yili Wang helps to verify the proposed method for multi-person pose estimation tasks at the early stage.