Can We Supervise Decoder Stages Better?

DDA4220 Final Project

Lai Wei (120090485)
School of Data Science
Chinese University of Hong Kong, Shenzhen
laiwei1@link.cuhk.edu.cn

Huihan Yang (120090438)
School of Management and Economics
Chinese University of Hong Kong, Shenzhen
huihanyang@link.cuhk.edu.cn

Rongxiao Qu (120020144)
School of Data Science
Chinese University of Hong Kong, Shenzhen
rongxiaoqu@link.cuhk.edu.cn

Abstract

In this paper, we examine the Deformable DETR framework's performance limitations on VOC2012 dataset in terms of the stage performance by the quantized measurements and visualization tools. We try to improve the performance by supervising decoder stages differently. We reveal that auxiliary loss is effective, but it also contains the stage-wise performance side-effects. Then, we propose two approaches to address the issues. We introduce and implement a re-weighted auxiliary loss strategy and Selective Query Recollection (SQR), a method using selective query information to improve object detection. Experiments on benchmark dataset show an improvement of up to 2.0% in mAP50. Our study contributes to a better understanding of the stage supervision in Deformable DETR and offers practical solutions for enhancing performance. Our findings may be also used to other transformer-based models. The code is available at `https://github.com/CUHKSZ-RMZHANG/final_project-I-am-Future`.

1 Workload

- Lai Wei (33.4%): Dataset preparation, Code implementation, Experiments, Report.
- Huihan Yang (33.3%): Experiments Design, Experiments, Report.
- Rongxiao Qu (33.3%): Observation experiments, Visualization, Report.

2 Introduction

In recent years, the field of object detection has made remarkable advancements with the transformer architectures. The detection transformers [1] have garnered significant attention for their novel designs to accurately identify and locate objects within images. A common transformer-based detector consists of the feature extraction backbone, the transformer encoder and decoder, and the MLP head. The typical decoder structure is shown in Fig. 1. The queries passes through a stack of decoder layers to predict the categories and bounding boxes. During the propagation, the intermediate output is also used for supervision, where the corresponding loss is known as the auxiliary loss.

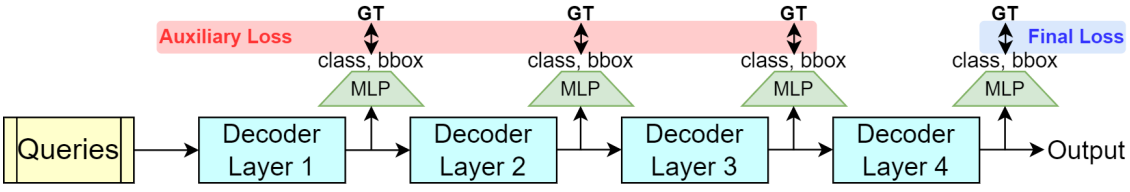

Figure 1: Typical structure of the detection transformer decoder (4 layers)

We have done experiments to check the effectiveness of the decoder stage supervision from auxiliary loss (See Table 3). The auxiliary loss indeed could bring about 4% mAP improvement on the VOC2012 dataset. However, we think that the properties of the auxiliary loss may limit the model's performance. To further investigate the effects of auxiliary loss, we inspect the model's stage-wise prediction. Some examples are shown in Fig. 2.

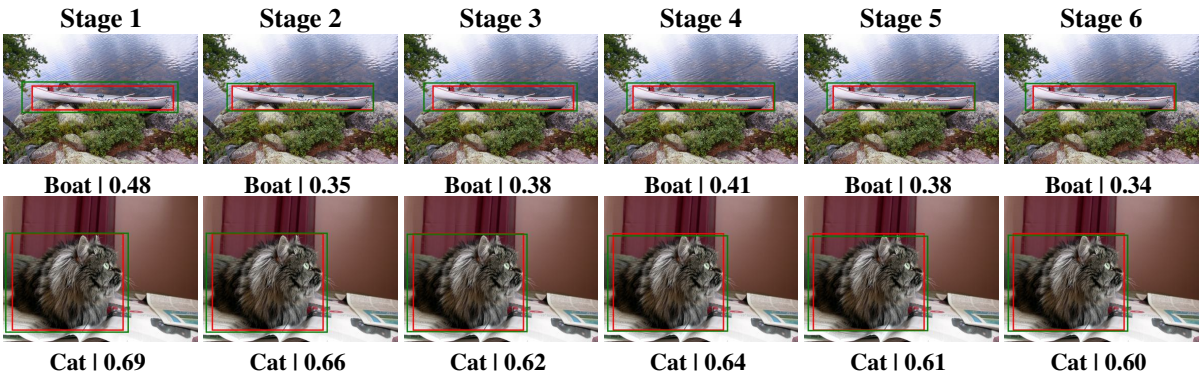

Figure 2: Stagewise predictions of two examples. The red box is the ground truth, and the green box is the model prediction. The classification category with confidence score is shown below each figure.

We observed that despite overall detection accuracy improving, there are many instances where the last layer's output performs worse than the intermediate layers. As examples shown in figure 2, even though the first layers of the deformable DETR's transformer's output have already recognized the object with high confidence, the final output results are often inferior.

3 Related Work

For decades, the object detection community mainly have three branches, the multi-stage detectors [2, 3, 4], the single-stage detectors [5, 6, 7] and the transformer based detectors [1]. For the first two types of models, they utilize dense priors such as anchor boxes or anchor points to match ground-truth (GT) objects based on their Intersection-over-Union (IoU) values or other sophisticated scoring factors. In recent years, DETR and its variants [1, 8, 9] introduced a new transformer-based object detection method with an end-to-end training strategy, leveraging self-attention mechanisms and the Hungarian Matching algorithm for object query matching. Since the new design doesn't need any human priors to the tasks, it has attracted many researchers' interests.

To address the slow convergence of the detection transformer, an auxiliary loss is applied to intermediate transformer decoding layers. Deformable DETR [8] incorporates a deformable attention module, enabling adaptive attention sampling locations, accelerating convergence, and simplifying the detection pipeline by eliminating anchor boxes and region proposals. Conditional DETR decouples object queries into content and spatial queries, enabling faster learning of object extremities.

Selective Query Recollection (SQR) [10] mitigates issues where detectors correctly predict at intermediate stages but mispredict at the final decoding stage by accumulating and selectively forwarding intermediate queries to later stages. This process emphasizes later stages, reduces errors, and optimizes computational load, ultimately improving detection precision.

4 Method

4.1 Baseline

The VOC2012 Dataset [11] contains 5717 training images and 5823 labeled validation images. We randomly split the given validation set of two sets, one contains 2911 images as the validation set during the training phase, and the other one contains 2912 images as the testing set to measure the final model performance.

In this study, we examine various experimental configurations to establish a baseline performance metric. Our investigation incorporates the original DEtection TRansformer (DETR) model [1] using its official implementation [12], as well as the Deformable DETR [8] with its official implementation [13]. We observed that employing the paper's suggested setting of `num_queries=100` for the DETR model proved challenging in terms of convergence. This difficulty arises due to the disparity in the average number of objects per image between VOC2012 Dataset [11] (2.38 objects) and MS COCO Dataset [14] (7.19 objects). Consequently, we opted for `num_queries=25` for the DETR model. Regarding the Deformable DETR, we experimented with `num_queries` values of 40, 75, and 150, considering the original recommendation of `num_queries=300`. The mean Average Precision (mAP) on the final test dataset and the training duration (measured in GPU hours)[1] are presented in Table 1.

From the Table 1, we find that Deformable DETR has higher training speed and higher performance compared to the DETR. Therefore, we choose the Deformable DETR as our model in the future experiments. Among the Deformable DETR experiments, we found that the overall performance increases as the `num_queries` increases.

[1] In this report, 1 GPU hour denotes training on an RTX 3090 GPU for one hour.

Table 1: Comparison among different baseline settings

Model	num_queries	mAP50	Training GPU hours
DETR	25	64.7	70
Deformable DETR	40	66.5	16
Deformable DETR	75	66.5	20
Deformable DETR	150	67.9	32

But the training GPU hours significantly increase as well. Finally, we balance the options and decide to choose the Deformable DETR with `num_queries=75` as our baseline.

4.2 Improvement: Re-weighting the loss

The auxiliary loss supervised the earlier decoder layers converge, therefore the performance increases. However, an obvious intuition from this supervision is that it would "mislead" the later stage's prediction. As shown from Fig. 1, auxiliary loss will guide the earlier decoder layers to learn the whole information needed for the MLP to do the predictions. In that way, the later stages could learn nothing and would degrade as identity functions. To suppress this phenomenon, one straight-forward idea is to re-arrange the supervision by re-weighting the auxiliary losses imposed on each layers. The earlier layers should have a smaller weight, so that auxiliary loss can merely guide the earlier layers a little bit to the correct directions, which would be beneficial for training.

We conduct experiments from the different weight distributions to the five intermediate stages. The first test shutdown all of the auxiliary losses (i.e., {0,0,0,0,0} for all intermediate stages). The second is linear weights {1/6, 2/6, 3/6, 4/6, 5/6}. And the last test is inspired by the Fibonacci sequence (1,2,3,5,8,13, ...), with the weights {1/13, 2/13, 3/13, 5/13, 8/13}.

4.3 Improvement: Selective Query Recollection(SQR)

We look more carefully in the structure of decoder stage supervision. We find that the sequential decoder structure could cause potential cascading errors. A refined intermediate query, irrespective of its positive or negative impact, will cascade to subsequent stages, while the unrefined query, which could potentially be more representative, is not given the chance to propagate forward, remaining inaccessible to the later stages.

Based on the identified issues, we adopt Query Recollection (QR) training strategy[10] on Deformable DETR to improve decoder stages supervision. QR accumulates intermediate queries as stages progress and provides them to downstream stages, bypassing the sequential structure. At each stage, new and original inputs are treated independently, with individual attentions and losses calculated.

Also,considering the fact that early stages' query may not be suitable to be recollected in the later stage due to the existing large gap. We recollect queries from the adjacent stage and the stage prior to the adjacent stage, which are more likely to bring positive effects.

Therefore, we adopt the Selective Query Recollection(SQR) training strategy as illustrated in Fig.3. Denote D_s as starting stage, C^s as query collection at stage s. During the flow of queries, queries are recollected with formula

$$C^0 = \{q^0\},\ C^1 = \{q^0, q^{01}\},\ C^s = \{D^s(q)|q \in C^{s-1}\} \bigcup \{D^{s-1}(q)|q \in C^{s-2}\} \tag{1}$$

Moreover, it is unnecessary to recollect queries from early stage with results showed from experiments. Pushing the starting stage later could reduce computational cost. More specifically, we conduct experiments from the different starting stages as stage 1 (query collection {1,2,3,5,8,13}, 32 supervisions in total), stage 2 (query collection {1,1,2,3,5,8}, 20 supervisions in total) and staged 3 (query collection {1,1,1,2,3,5}, 13 supervisions in total).

We also conduct experiments on combining loss reweighting method and SQR method. We use stage auxiliary loss weight as Fibonacci sequence and SQR with stage 1 as starting stage.

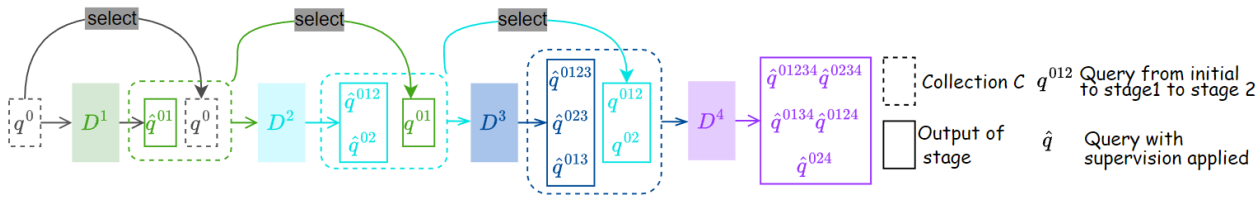

Figure 3: Selective Query Recollection (4 layers decoder

5 Experiments

5.1 Evaluation method

We use mAP50, TP Fading rate and FP Exacerbation rate to measure the performance.

mAP50 is adopted to measure the detection accuracy of models. We use coco evaluator [15] to measure it.

$$AP50 = \frac{TP}{TP + FP},\ mAP50 = \frac{1}{K}\Sigma_{k=1}^{K} AP50_k \tag{2}$$

where the threshold of bounding box's IoU is 50%, $AP50_k$ represents k^{th} class's AP and K represents number of total classes.

TP Fading rate:If P_i^6 is a true-positive (TP) towards a ground-truth G, we check whether $P_i^{1\sim5}$ generate a better TP towards the same G that has higher IoU & higher category score than P_i^6. The occurrence rate is denoted as TP fading rate.

FP Exacerbation rate: If P_i^6 is a false-positive (FP), we check whether $P_i^{1\sim5}$ produce a FP but with lower category score than P_i^6. The occurrence rate is denoted as FP exacerbation rate.

P_i^k denotes predicted results of query i on stage k. Details can be found in Appendix A.

5.2 Experimental details

We clone the code from [13] and implement the improvement methods by ourselves. Since the transformer architecture is memory-consuming, we use a high performance Linux server equipped with eight RTX3090 GPUs for training. General experimental setups are adopted to every experiments.

Table 2: General Experimental Setups

Batch size	Learning rate	Lr annealing	Num epoch	Num queries
2 / GPU	1×10^{-4}	$\gamma = 0.1$@epoch40	65	75

* Other settings are the same as original Deformable DETR.

Motivation. The original implementation trains with 8 GPUs · 2 samples per GPU. In our experiments, we use 4 GPUs · 2 samples per GPU. Therefore, the learning rate should be halved to 1×10^{-4}. We decay the learning rate to 0.1 times at epoch 40 epoch to refine the model. Number of epoch is selected according to the performance of loss, which converges at around 50 epochs. The detailed discussion about query number is at Sec. 4.1.

5.3 Results and analysis

Table.3 records our experiments' numerical results. Note that we train our model from scratch on VOC2012 dataset, the performance may not be good as the one uses pretrained models.

It shows that varying supervision (emphasis on late stages) can be useful on improving model's performance. The decrease of TP F rate across two thresholds indicates the alleviation of stage-wise problem. The increase of AP50 shows the model's increasing accuracy on detection. This improvement verifies our doubts on original supervision and proves the efficiency of our methods.

Table 3: Final experiment results.

Model	Training GPU hours	mAP50↑	IoU≥0.5 TP F rate	IoU≥0.5 FP E rate	IoU≥0.75 TP F rate	IoU≥0.75 FP E rate
D-DETR[a]	20	66.5	47.9%	78.2%	44.4%	75.7%
D-DETR w/o aux loss	18	62.3 (-4.2)	0.1%	36.7%	0.0%	3.9%
D-DETR linear[b]	20	68.1 (+1.6)	40.5%	79.1%	35.7%	71.8%
D-DETR fibonacci	20	**68.5** (+2.0)	33.9%	80.3%	28.7%	72.6%
SQR D-DETR s13[c]	24	67.3 (+0.8)	41.4%	76.9%	37.4%	73.6%
SQR D-DETR s20	28	**68.2** (+1.7)	43.3%	78.6%	38.6%	78.8%
SQR D-DETR s32	36	67.8 (+1.3)	42.9%	72.3%	40.7%	62.3%
SQR D-DETR s32 fibonacci	36	67.1 (+0.6)	42.5%	79.6%	40.3%	66.2%

a. The "D-DETR" is the shorthanded version of Deformable DETR.
b. The "linear" and "fibonacci" indicate the use of linear and Fibonacci sequence stage weights.
c. The "s13, s20, s32" means the number of supervisions, corresponds to the recollection begining at decoder 1, 2, 3 respectively.

5.3.1 Analysis: Loss Reweighting

In the experiment of removing auxiliary loss, we find that the mAP decreases for 4%, indicating the auxiliary supervision is effective. Also, TP Fading rate is almost 0%, which means that almost all the detected objects' confidence increases from stage to stage. This is because model is only supervised in the last stage. Last stage should behave better then the before. We could also conclude that Auxiliary losses lead to high TP Fading rates.

Changing the supervision by reweighting the loss is efficient with increasing mAP50(+1.6 ~2.0) and decreasing the TP Fading Rate. It could further alleviate the stage-wise performance problem and improve model's detection accuracy. This is because by paying more focus on later stages, the model is forced to be better on late stages. It indicates training with emphasis on late stages is a better option.

However, it turns that the loss reweighting have higher FP Exacerbation Rate compared with FP Exacerbation Rate from SQR. This indicates that access to the intermediate queries is necessary for emphasized supervision. Only using loss reweighting techniques, the model may have bias in recognizing some objects.

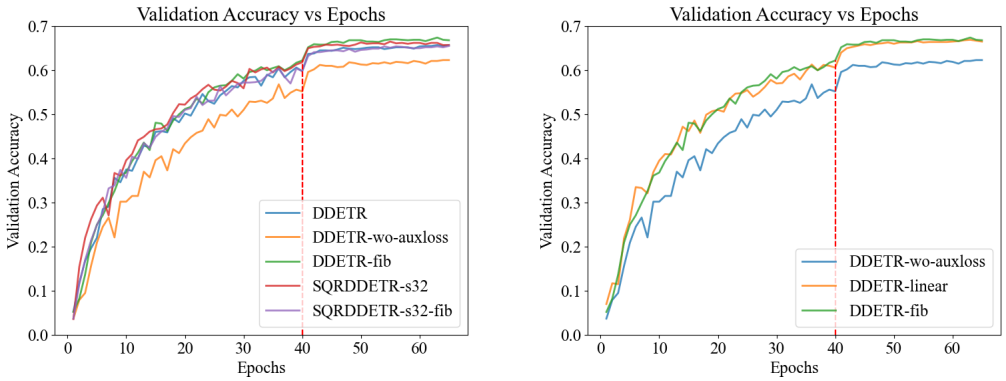

Figure 4: Validation Accuracy of Different Settings. The red dash line indicate the learning rate decay.

5.3.2 Analysis: SQR

SQR also improves the model's performance significantly by adjusting supervision. It not only increases model's detection accuracy(+0.8 ~1.7) but also decreases both TP Fading Rate and FP Exacerbation Rate. This is mainly because by using query recollection, we implicitly emphasizes the supervision on late stages and connects to intermediate queries. By adjusting the supervision, we force model to be more accurate in the late stages. By properly connecting with intermediate queries, we strengthen the correlation and optimization between stages.

Moreover, the best performance is achieved when starting query collection at stage 2 with efficiency. This is mainly because there exists learning gap between the stages and early query which combining them may lead to bad performance. Moreover, less query amount recollected also reduces the training time.

It is also worth to notice that during training, SQR tends to be converged faster and varies more than DETR as Fig.4. This is mainly due to the more freedom given to the intermediate queries.

5.3.3 Analysis: Combination of SQR & Loss Reweighting

We try to achieve a better performance by combining the two methods mentioned above. However, it's interesting to find that simply combination only leads to the offset of two methods' advantages.

This is mainly because two methods improve the performance in two different ways. The loss reweighting guide the later stage to learn the residual information from the previous stage. And the SQR guide the earlier stage to learn more generalized knowledge. Also, by combining these two methods, we loose the supervision on early stages twice, which may leads to poor performance from early stages. Due to the sequential structure of decode, bad performance from early stages could have some impact on later stages.

6 Conclusion

In this paper, we investigate the stage-wise performance issue in the Deformable DETR framework when applied to the VOC2012 dataset. Our findings indicates that the existing stage-wise supervision is effective, but it contains side effect that misleading the model's later stages prediction. To tackle this issue, we use two approaches, re-weighting the loss and Selective Query Recollection, to improve the overall performance of the model.

Our experiments show that the re-weighting of the loss function with a linear or Fibonacci sequence-based distribution of weights effectively enhances the model's performance by increasing the mAP50 and reducing the TP Fading Rate. Moreover, the SQR approach, which accumulates intermediate queries from adjacent stages and selectively propagates them to the downstream stages, is proved to be effective in mitigating cascading errors and improving model performance.

Our work may shed light on other strategies for refining intermediate supervision signals and further optimize the model's performance on various object detection tasks.

Acknowledgement

The authors gratefully acknowledge the high performance computing resources from Prof. Xiaoguang Han (SSE).

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[3] Ross Girshick. Fast r-cnn, 2015.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[6] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.

[7] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang. Tood: Task-aligned one-stage object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3490–3499, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

[8] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection, 2021.

[9] Shilong Liu, Tianhe Ren, Jiayu Chen, Zhaoyang Zeng, Hao Zhang, Feng Li, Hongyang Li, Jun Huang, Hang Su, Jun Zhu, and Lei Zhang. Detection transformer with stable matching, 2023.

[10] Fangyi Chen, Han Zhang, Kai Hu, Yu kai Huang, Chenchen Zhu, and Marios Savvides. Enhanced training of query-based object detection via selective query recollection, 2023.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[12] Facebook Research. Detr. `https://github.com/facebookresearch/detr`.

[13] SenseTime. Defomrable-detr. `https://github.com/fundamentalvision/Deformable-DETR`.

[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[15] Yuxin Wu. pycocotools. `https://github.com/ppwwyyxx/cocoapi`.

[16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Appendix A: TP Fading rate and FP Exacerbation rate explanation

From the work [10], queries are updated successively. An initial query $q_i^0 (i \in N = \{1, 2, \ldots, n\})$ is an embedding trained through back-propagation, and n is the total number of initial queries. During inference, the first stage updates the initial query by adding a residual term to it, producing intermediate query q_i^1, followed by later stages sequentially updating the intermediate query in the same way. The procedure can be formulated as

$$q_i^s = D^s \left(q_i^{s-1}, q^{s-1}, x\right) = q_i^{s-1} + (\mathcal{A} \circ \mathcal{F}) \left(q_i^{s-1}, q^{s-1}, x\right) \tag{3}$$

D^s is a decoding stage where s is stage index; q_i^s is the i_{th} query at stage s; q^s is a set of queries $q^s = \{q_i^s \mid i \in N\}$; $(\mathcal{A} \circ \mathcal{F})$ stands for a bundle of modules including self and cross attention and feed forward network; x means

features; and LayerNorm [16] that applied on each module is omitted for simplicity. Afterward, q_i^s predicts an object P_i^s via two multi-layer perceptrons for classification and regression:

$$P_i^s = \left(MLP_{cls}\left(q_i^s\right), \quad MLP_{reg}\left(q_i^s\right)\right) \tag{4}$$

$P_i^{1\sim6}$ are predicted by the $q_i^{1\sim6}$ rooted in q_i^0, where P_i^6 is the expected outcome and $P_i^{1\sim5}$ are intermediate results. P_i^s is regarded as a true-positive towards a ground-truth G only if the $\mathrm{IoU}\left(P_i^s, \mathrm{G}\right)$ exceeds a threshold, its category matches with G, and the categorical score is ranked as the highest among all other counterparts.

The definition for the **TP Fading rate** and **FP Exacerbation rate**:

(1) If P_i^6 is a true-positive (TP) towards a ground-truth G, we check whether $P_i^{1\sim5}$ generate a better TP towards the same G that has higher IoU & higher category score than P_i^6. The occurrence rate is denoted as TP fading rate.

(2) If P_i^6 is a false-positive (FP), we check whether $P_i^{1\sim5}$ produce a FP but with lower category score than P_i^6. The occurrence rate is denoted as FP exacerbation rate.

Appendix B: Code explanation

The code repository is submitted to the Github at `https://github.com/CUHKSZ-RMZHANG/final_project-I-am-Future`, where all components, including the data preparation scripts, the main Deformable DETR code, and post analysing scripts exist. Follow the `README.md` to setup the environment and run the code.

The training log, testing log, and the trained weights of all of our experiment is available at `https://cuhko365-my.sharepoint.com/:f:/g/personal/120090485_link_cuhk_edu_cn/Eq_qt23vsLJAi-eJNlTUhjwBzmT139TiyoKf7vEnoCKoDg?e=3wCdPa`.

Our pre-processed COCO format VOC2012 dataset is available at `https://cuhko365-my.sharepoint.com/:u:/g/personal/120090485_link_cuhk_edu_cn/EXF-XYXpeUhDso64knZbP2cB7bz6snA8dMPKfR16ANYe1Q?e=3neaZO`.

If there are any problems reproducing the code, or the OneDrive link is dead, please contact `laiwei1@link.cuhk.edu.cn`.