DEIM: DETR with Improved Matching for Fast Convergence

Shihua Huang; Zhichao Lu; Xiaodong Cun; Yongjun Yu; Xiao Zhou; Xi Shen

Intellindust AI Lab, City University of Hong Kong, Great Bay University, Hefei Normal University

Feb, 2025

Presented by Scott Sun from Duke B&B

Introduction

DETR (DEtection TRansformer) [1] is an object detection model that reformulates detection as a direct set prediction problem using a Transformer encoder–decoder architecture. Instead of relying on anchors, region proposals, or non-maximum suppression (NMS), DETR predicts a fixed-size set of object queries in parallel. *DEIM [2] is an innovative and efficient training framework designed to accelerate convergence in real-time object detection.*

Goal: accelerate model convergence in set prediction (i.e., object detection) through: (1) **dense matching strategy (Dense O2O)** and (2) **matchability-aware loss (MAL)**

- **DETR:** Training follows an EM-like procedure. In the `E-like step`, bipartite matching via the Hungarian algorithm establishes a one-to-one assignment between predicted object queries and ground-truth objects, ensuring permutation-invariant set prediction. In the `M-step`, the model takes gradient update to min classification and localization losses based on the matched pairs.
- **real-time DETR (RT-DETR/D-FINE):** Accelerated DETR variants designed for real-time object detection, where the standard transformer encoder is replaced by hybrid encoders for better multi-scale representation learning.

Background: Hungarian Matching

The Hungarian method is a combinatorial optim algorithm solving the assignment problem in $O(n^3)$. The one-to-one matching can be easily done through `scipy.optmize.linear_sum_assignment`.

Worker \ Task	Clean bathroom	Sweep floors	Wash windows
Alice	**$8**	$4	$7
Bob	$5	$2	**$3**
Carol	$9	**$4**	$8

Figure: Matching cost matrix in toy example for Hungarian Algorithm

In DETR, queries (i.e. placeholder) are assigned to ground truth objects. Here, $n = \#$queries (i.e. 100). The cost matrix is more complex. More formally, in Hungarian matching, we search for a optimal permutation of n elements, $\sigma \in \mathcal{P}_n$ s.t.

$$\hat{\sigma} = \underset{\sigma \in \mathcal{P}_n}{\arg\min} \sum_{i}^{n} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) \tag{1}$$

$$\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)}) \tag{2}$$

Background: Hungarian Matching in DETR

Here is a demonstration of how we apply Hungarian matching on a batch of samples. The matching step is performed on the CPU, and the ncol can vary dynamically. Note that substantial padding is currently used p, leaving significant room for algorithmic optimization and innovation.

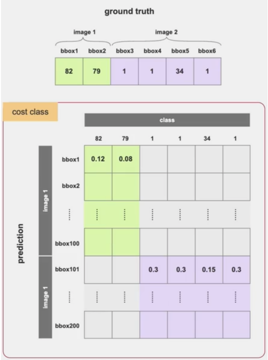


Figure: Illustration of the cost matrix for a batch containing two images (i.e., two indep set pred problems). Of note, this is the reduced matrix, where we ignore the matching for ∅.

Background: DETR

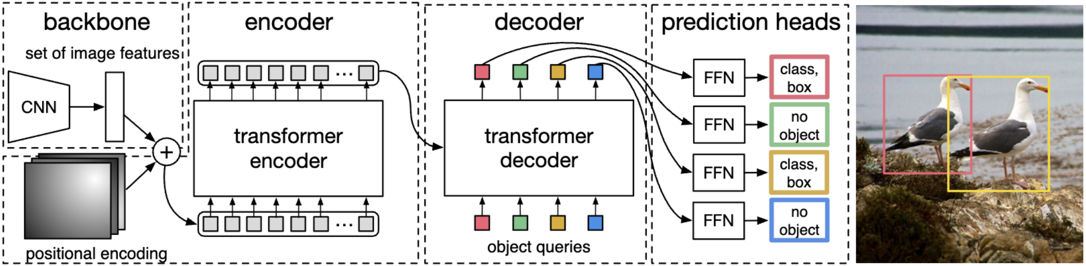


Figure: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call object queries, and additionally attends to the encoder output. We pass each output embedding of the decoder to two shared FFNs (one for classification, and the other one for box localization).

Background: Classification Losses

The evolution of classification losses used in DETR and its successors (calculated at the instance level). Let y be ground truth class.

- DETR CE loss over softmax probs (rank all class probs):

$$\mathrm{CE} = -\log(p_y)$$

- Since Deformable DETR, class probs are switched independent sigmoid class probs (indep classify whether or not the query of interest is object c for $\forall c \in \mathcal{C}$):

Background: Classification Losses

BCE $\rightarrow$ Focal Loss (FL) $\rightarrow$ VariFocal Loss (VFL) $\rightarrow$ MAL (which is proposed by DEIM). Let p be the predicted probability, y be the binary label (0 or 1), α be the balancing factor, γ be the focusing parameter, and q be the IoU btw predicted bbox and tre bbox.

- BCE:

$$\mathrm{BCE}(p, y) = \begin{cases} -\log(p) & y = 1 \\ -\log(1-p) & y = 0, \end{cases}$$

- FL:

$$\mathrm{FL}(p, y) = \begin{cases} -\alpha(1-p)^{\gamma}\log(p) & y = 1 \\ -(1-\alpha)p^{\gamma}\log(1-p) & y = 0, \end{cases}$$

- VFL:

$$\mathrm{VFL}(p, q, y) = \begin{cases} -q(q\log(p) + (1-q)\log(1-p)) & q > 0 \\ -\alpha p^{\gamma}\log(1-p) & q = 0, \end{cases}$$

Method: DETR Limitation

Major issue of DETR: both training and inference are computational expensive. It takes $\sim$500 epochs to reach convergence on the COCO dataset (330K images with avg size 640×480).

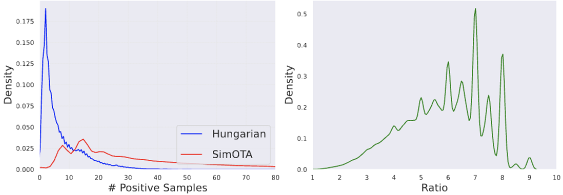
- insufficient supervision in O2O matching
- low quality matching

These are the critical limitations that cause slow convergence in DETR and its real-time variants.

Method: O2M vs. O2O

Traditional YOLO-based methods typical use O2M matching strategy. This approach enable dense supervision (many positive samples) and thus allow for higher recall (true pos). But, the downside is that we need handcrafted NMS to refine bboxes...

DETR eliminate NMS via sparse O2O matching via the Hungarian Algorithm. But, the downside is that we may not have enough supervision, which leads to slow convergence...

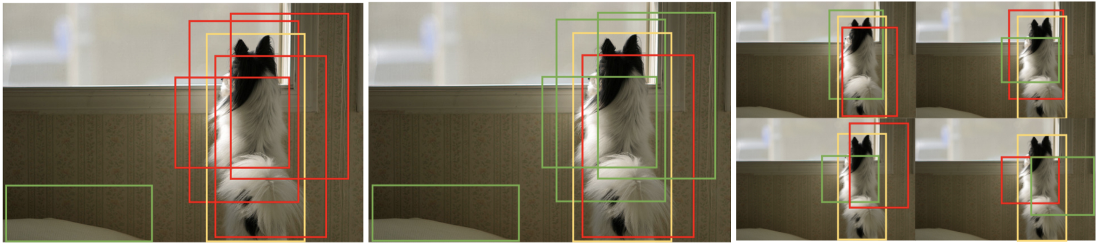


(a) Matching distribution (b) Ratios between O2M and O2O

Figure: O2M vs O2O: anchor/query match per image using COCO. SimOTA is a O2M strategy.

This demonstrates that O2O has fewer positive matches, potentially slowing down optimization

Method: Dense O2O



(a) **O2M**: 1 target and 4 pos. (b) **O2O**: 1 target and 1 pos. (c) **Dense O2O by stitching**: 4 targets and 4 pos.

Figure: Dense-O2O = stitching (mosaic) / mixup [1]

Mosaic augmentation = combines four different training images into one single image in a 2x2 grid
MixUp augmentation = 0.5 image A + 0.5 image B

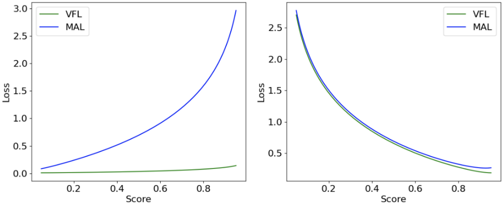
the goal is to create more positive matching for more supervision

[1] the actual implementation has a complex augmentation scheduling

Method: Matchability-Aware Loss (MAL)

People use VFL in SoTA DETR variants for classification, but VFL has two limitations: 1) unaware of low-quality matches, 2) treating matches with 0 IoU as negative samples, which reduce the number of effective positive signals.

$$\mathrm{MAL}(p, q, y) = \begin{cases} -q^{\gamma} \log(p) - (1 - q^{\gamma}) \log(1 - p) & y = 1 \\ -p^{\gamma} \log(1 - p) & y = 0. \end{cases} \tag{3}$$



(a) **Low quality**: IoU = 0.05 (b) **High quality**: IoU = 0.95

Figure: MAL vs. VFL in cases of low-quality matching and high-quality matching. Score is p. MAL has a much stronger signal than VFL when there is a confident low quality matching.

Experiment: Comparisons with real-time detectors using COCO

Table 1. **Comparison with real-time object detectors on COCO [20] val2017.** By integrating our method into D-FINE-L [27] and D-FINE-X [27], we build DEIM-D-FINE-L and DEIM-D-FINE-X. We compare our method with YOLO-based and DETR-based real-time object detectors. ⋆ indicates that the NMS is tuned with a confidence threshold of 0.01.

Model	#Epochs	#Params	GFLOPs	Latency (ms)	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLO-based Real-time Object Detectors										
YOLOv8-L [12]	500	43	165	12.31	52.9	69.8	57.5	35.3	58.3	69.8
YOLOv8-X [12]	500	68	257	16.59	53.9	71.0	58.7	35.7	59.3	70.7
YOLOv9-C [34]	500	25	102	10.66	53.0	70.2	57.8	36.2	58.5	69.3
YOLOv9-E [34]	500	57	189	20.53	55.6	72.8	60.6	40.2	61.0	71.4
Gold-YOLO-L [33]	300	75	152	9.21	53.3	70.9	-	33.8	58.9	69.9
YOLOv10-L⋆ [32]	500	24	120	7.66	53.2	70.1	58.1	35.8	58.5	69.4
YOLOv10-X⋆ [32]	500	30	160	10.74	54.4	71.3	59.3	37.0	59.8	70.9
YOLO11-L⋆ [13]	500	25	87	6.31	52.9	69.4	57.7	35.2	58.7	68.8
YOLO11-X⋆ [13]	500	57	195	10.52	54.1	70.8	58.9	37.0	59.2	69.7
DETR-based Real-time Object Detectors										
RT-DETR-HG-L [43]	72	32	107	8.77	53.0	71.7	57.3	34.6	57.4	71.2
RT-DETR-HG-X [43]	72	67	234	13.51	54.8	73.1	59.4	35.7	59.6	72.9
D-FINE-L [27]	72	31	91	8.07	54.0	71.6	58.4	36.5	58.0	71.9
DEIM-D-FINE-L	**50**	31	91	8.07	**54.7**	72.4	59.4	36.9	59.6	71.8
D-FINE-X [27]	72	62	202	12.89	55.8	73.7	60.2	37.3	60.5	73.4
DEIM-D-FINE-X	**50**	62	202	12.89	**56.5**	74.0	61.5	38.8	61.4	74.2

Experiment: Comparisons with ResNet-based DETR using COCO

Table 2. **Comparison with ResNet-based DETRs on COCO [20] val2017.** By integrating our method into ResNet50 [14] and ResNet101 [14], we build DEIM-RT-DETRv2-R50 and DEIM-RT-DETRv2-R101. We compare our method with competitive DETR-based object detectors that use ResNet50 [14] or ResNet101 [14] as backbones.

Model	#Epochs	#Params	GFLOPs	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
ResNet50 [14]-based									
DETR-DC5 [3]	500	41	187	43.3	63.1	45.9	22.5	47.3	61.1
Anchor-DETR-DC5 [35]	50	39	172	44.2	64.7	47.5	24.7	48.2	60.6
Conditional-DETR-DC5 [26]	108	44	195	45.1	65.4	48.5	25.3	49.0	62.2
Efficient-DETR [36]	36	35	210	45.1	63.1	49.1	28.3	48.4	59.0
SMCA-DETR [11]	108	40	152	45.6	65.5	49.1	25.9	49.3	62.6
Deformable-DETR [45]	50	40	173	46.2	65.2	50.0	28.8	49.2	61.7
DAB-Deformable-DETR [21]	50	48	195	46.9	66.0	50.8	30.1	50.4	62.5
DN-Deformable-DETR [18]	50	48	195	48.6	67.4	52.7	31.0	52.0	63.7
DINO-Deformable-DETR [39]	36	47	279	50.9	69.0	55.3	34.6	54.1	64.6
RT-DETR [43]	72	42	136	53.1	71.3	57.7	34.8	58.0	70.0
RT-DETRv2 [24]	72	42	136	53.4	71.6	57.4	36.1	57.9	70.8
DEIM-RT-DETRv2	**36**	42	136	53.9	71.7	58.6	36.7	58.9	70.9
DEIM-RT-DETRv2	**60**	42	136	**54.3**	72.3	58.8	37.5	58.7	70.8
ResNet101 [14]-based									
DETR-DC5 [3]	500	60	253	44.9	64.7	47.7	23.7	49.5	62.3
Anchor-DETR-DC5 [35]	50	-	-	45.1	65.7	48.8	25.8	49.4	61.6
Conditional-DETR-DC5 [26]	108	63	262	45.9	66.8	49.5	27.2	50.3	63.3
Efficient-DETR [36]	36	54	289	45.7	64.1	49.5	28.2	49.1	60.2
SMCA-DETR [11]	108	58	218	46.3	66.6	50.2	27.2	50.5	63.2
RT-DETR [43]	72	76	259	54.3	72.7	58.6	36.0	58.8	72.1
RT-DETRv2 [24]	72	76	259	54.3	72.8	58.8	35.8	58.8	72.1
DEIM-RT-DETRv2	**36**	76	259	55.2	73.3	59.9	37.8	59.6	72.8
DEIM-RT-DETRv2	**60**	76	259	**55.5**	73.5	60.3	37.9	59.9	73.0

Recommendation

Is it worth reading? Yes, but DETR is more recommended as it's a foundational work.

- It gives detailed background/related works
- Many techniques are exclusively designed for object detection

Is it worth implementing? Yes. DETR is a good starting point, but it needs modification to adapt my use case.

- It is worthwhile to implement the Hungarian Matching algorithm for set/mset prediction in my use case

References I

[1] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[2] S. Huang, Z. Lu, X. Cun, Y. Yu, X. Zhou, and X. Shen. Deim: Detr with improved matching for fast convergence. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15162–15171, 2025.