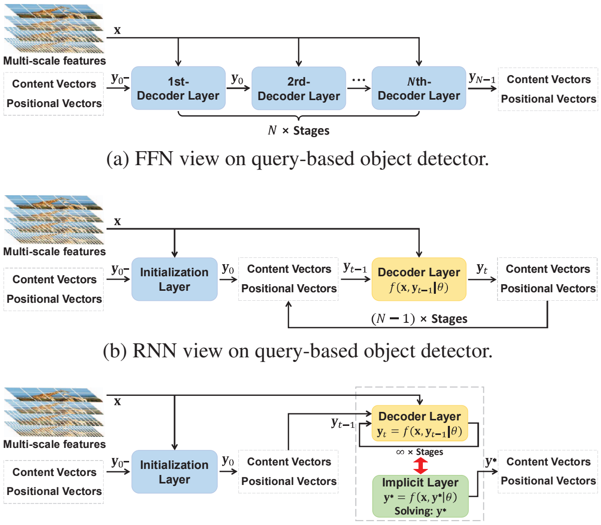
Deep Equilibrium Object Detection

Shuai Wang[1] Yao Teng[1] Limin Wang[1,2,✉]
[1]State Key Laboratory for Novel Software Technology, Nanjing University [2]Shanghai AI Lab
https://github.com/MCG-NJU/DEQDet

Abstract

Query-based object detectors directly decode image features into object instances with a set of learnable queries. These query vectors are progressively refined to stable meaningful representations through a sequence of decoder layers, and then used to directly predict object locations and categories with simple FFN heads. In this paper, we present a new query-based object detector (DEQDet) by designing a deep equilibrium decoder. Our DEQ decoder models the query vector refinement as the fixed point solving of an **implicit** *layer and is equivalent to applying* **infinite** *steps of refinement. To be more specific to object decoding, we use a two-step unrolled equilibrium equation to explicitly capture the query vector refinement. Accordingly, we are able to incorporate refinement awareness into the DEQ training with the inexact gradient back-propagation (RAG). In addition, to stabilize the training of our DEQDet and improve its generalization ability, we devise the deep supervision scheme on the optimization path of DEQ with refinement-aware perturbation (RAP). Our experiments demonstrate DEQDet converges faster, consumes less memory, and achieves better results than the baseline counterpart (AdaMixer). In particular, our DEQDet with ResNet50 backbone and 300 queries achieves the* 49.5 *mAP and* 33.0 *AP$_s$ on the MS COCO benchmark under* 2× *training scheme (24 epochs).*



(a) FFN view on query-based object detector.

(b) RNN view on query-based object detector.

(c) Our DEQDet view on query-based object detector.

Figure 1: **Different views on query-based object detector.** (a) In FFN view, the decoder consists of stacked non-shared decoder layers *e.g.* AdaMixer [14] (b) In RNN view, the decoder consists of weight-tied decoder layers. (c) In our DEQDet, decoder performs refinement in a RNN manner, but model it as the fixed point solving of an implicit layer with infinite steps. The red arrow means equivalence.

1. Introduction

Object detection [33, 24, 6, 35, 14] is a fundamental task in computer vision research. Its purpose is to identify the locations and categories of all object instances in an image. This task is challenging because object detector is usually required to deal with large variations in object instances. Traditional object detectors often make dense predictions based on a large quantity of candidates such as anchor boxes [27, 33, 5] or reference points [11, 44, 20]. Among these models, one-stage object detectors [27, 38, 24, 45] directly classify the candidates and regress the bounding boxes based on them. In addition, two-stage detectors [33, 16] adopt an additional initialization step to select a set of coarse object proposals from dense anchors, and then *refine* their locations and predict their categories. These two types of detectors often require hand-crafted post-processing techniques such as NMS to yield the final detection results.

Recently, query-based object detectors [6, 35, 14] present a new paradigm for object detection. As shown in Fig. 1a, the detectors are composed of a set of learnable object queries and several stacked non-shared decoder layers (e.g. cross-attention [6], dynamic convolution [35], dynamic MLPMixer [14]). In such a detector, these query vectors

✉: Corresponding author (lmwang@nju.edu.cn).

are progressively refined by each decoder layer, where image features are sampled or attended and transformed based on each query. After several steps of refinements, these query vectors could be directly transformed into object predictions with a simple FFN head. The success of query-based object detectors yields a flexible and simple paradigm of directly decoding object instances from images without any dense assumption (e.g. dense anchors) or post processing.

Despite the great success achieved by query-based object detectors, some important issues on their design still remain. First, *parameter efficiency is an important issue for these detectors* [35, 14]. Each decoder layer performs the same task of query refinement but has its own parameters, which leads to the large numbers of parameters and makes them prone to overfitting. Second, *depth of refinement (decoder layers) is another critical factor in detector design.* Intuitively, increasing refinement depth would scale up the detector capacity and hopefully contribute to a better detection performance with a proper optimization method. To address these critical issues, we first come up with a new RNN perspective on the query vector update as shown in Fig. 1b, partially inspired by RNN-based optical flow estimation [36]. In this sense, we employ the same transformation in each decoder layer (known as weight tying), which provides a parameter-efficient way to scale up these query-based object detectors and could be viewed as a kind of regularization technique. Yet, this recurrent query refinement would yield significant computational and memory overhead due to the tracking of long hidden-state history in the BPTT algorithm [41]. In addition, it still needs to determine the number of decoder layers. Therefore, as shown in Fig. 1c, we further improve this weight-tying refinement to an extreme version, and model it as the fixed point solving process of an implicit layer with an efficient deep equilibrium model (DEQ) [3]. This DEQ view on query-based object detection is able to simultaneously reduce its model parameters and increase its refinement depth to an infinite level.

Specifically, in this paper, we present a new query-based object detector, termed as **DEQDet**, by designing a deep equilibrium decoder. To be more specific, it is based on the recent research of implicit models like DEQ [3] and the recent query-based detector of AdaMixer [14]. Different from the previous query-based detectors [6, 35, 14], the decoder of DEQDet only has two different decoder layers: an initialization layer and an implicit refinement layer. As shown in Fig. 1c, after the coarse object predictions are generated by the initialization layer, they are passed through this implicit refinement layer with infinite steps of iterations. The object query refinement is represented as *infinite-level fix-point solving process* of an *implicit layer*, which could be solved by any black-box solver and enjoy the analytical backward pass independent of forward pass trajectories. This fixed-point modeling perspective would share several advantages: (i) it would greatly scale up the modeling capacity of query-based detectors and is more flexible to deal with the large-variations of object instances. (ii) it would not rely on the traditional BPTT algorithms of RNN without storing the hidden states, which can save the memory overhead and make the training more efficient. (iii) it is a general modeling framework that could be applied to different query-based object detectors for detection performance improvement.

When training DEQDet, we find it is important to inject the refinement awareness (*i.e.*, the ability to perceive the operation that is performed iteratively) into its model parameter update. However, the commonly used methods for computing the gradients of implicit layers, such as Jacobian-free backpropagation (JFB) [12], lack the refinement awareness. To tackle this problem, we propose to solve the two-step unrolled equilibrium equation with two new designs: i) Refinement-Aware Gradient (RAG). Through analysis, we find the refinement awareness corresponds to the high-order terms of Neumann-series expansion of inverse Jacobian term [15], as they simulate the gradients propagated along the reverse of the solving path. Therefore, we impose the *refinement gradient term*, *i.e.*, second-order terms of Neumann-series expansion, into the gradients. ii) Refinement-Aware Perturbation (RAP). To further enhance the refinement awareness, we perturb the *fixed-point* solving path by injecting noise. Compared with merely adding Gaussian noise to each decoder layer, our perturbation can be transmitted continuously with the iterations. Thus, the deep supervision of query-based detectors on the perturbed solving path can encourage the refinement layer to be aware of its refinement nature.

We verify the effectiveness of our DEQDet framework on MS-COCO validation dataset by following the common practice. Our experiments demonstrate that DEQDet converges faster, consumes less memory, and achieves better results than the baseline counterpart (AdaMixer). In particular, our DEQDet with ResNet50 backbone and 300 queries achieves the 49.5 mAP and 33.0 AP_s on the MS COCO benchmark under 2× training scheme (24 epochs). We also perform in-depth ablation study on the design of DEQDet and verify its scaling performance with stronger backbones such as Swin-S. Our **contributions** are as follows:

- We introduce RNN view over the query-based object detector and propose to model it as a fixed-point of an implicit layer with infinite depth.
- We propose Refinement Aware Gradient for DEQ model applied in high level semantic understanding task with sparsity nature like object detection.
- We propose Refinement Aware Perturbation to simulate the real noise of *fixed-point* iterations in order to further improve refinement awareness of DEQ model.

- Our experiments demonstrate the DEQDet achieves the state-of-the-art performance under a fair setting on the MS-COCO dataset.

2. Related Work

Refinement in object detection. The framework of two-stage object detectors [33, 16] can be deemed as a refinement-based detection paradigm. In these detectors, an *initialization layer*, *e.g*., RPN [33], is first adopted to generate some proposals which provide rough locations of objects. Then, a *refinement layer* (*i.e*. the detection head formed by the region-wise feature extractor and the convolutional network) is employed to achieve precise localization and categorization for the object proposals. Multi-stage object detectors like Cascade R-CNN [5] introduce the *multi-step refinement* into object detection. Cascade R-CNN adopts a series of detection heads to gradually refine the bounding boxes of objects to enable the high quality detection. Query-based detectors [6, 46, 35, 14, 26, 13, 37] are proposed to perform object detection through a set of *learnable object queries*. These detectors are also formed by cascade decoder layers. In each layer, the image features are extracted by feature samplers and integrated into the input queries to generate the *intermediate representations*. These representations can not only serve as the input of the next layer for further refinement, but can also be decoded into the class labels and the bounding box coordinates [6] (or coordinate offsets) [35, 14] in current layer. Despite the great success of the query-based object detectors, these detectors are unable to guarantee the input and output intermediate representations of each layer to lay in the same latent space, because the decoder layers are not *weight-tied* [3]. This design is less parameter-efficient. In addition, it is hard to determine whether they has achieved the convergence of refinement. DiffusionDet [8] borrows the denoising training technique from diffusion models [19] into the refinement process of object detection, However, their refinement process is naively on the superficial space (formed by bounding boxes) instead of the latent representations.

Deep implicit neural network. Implict modeling has been explored by deep learning community by decades. Different from conventional neural networks that stack neural operators explicitly, implicit network defines its output by the solution of dynamic system. RBP [23, 31] trains the recurrent system implicitly by differentiation techniques. Neural ODE [7] employs black-box ODE solvers to model recursive residual block implicitly. Deep Equilibrium Model (DEQ) [3, 4, 12, 2, 39] defines an implicit layer of solving fixed point equation to corresponding to infinite depth. Our DEQDet aims to leverage this modeling power of implicit DEQ to the specific challenging object detection task and propose customized optimization techniques to improve its training effectiveness and efficiency for object detection.

3. Methodology

In general, our DEQDet can be applied to any query-based object detector. In current version, our DEQDet is mainly based on AdaMixer [14], a state-of-the-art query-based object detector which employs dynamic mixing in decoder design. We first present a brief introduction of AdaMixer. Then, in order to introduce our DEQDet, we present a RNN perspective to reveal the refinement nature of decoder layer. After that we formulate the detection decoder as a *fix-point* iteration process and propose our DEQDet. Finally we propose the training strategy of DEQDet.

3.1. AdaMixer Revisited

Given an image input $I \in R^{3\times H\times W}$, object detectors are required to output object bounding box and its corresponding class category. The query-based object detector use a backbone with or without a neck encoder to extract multi-scale image features $\mathbf{x} = \{x^1, x^2, ..., x^l\}$, where $x^i \in R^{D\times H^i\times W^i}$ is the i-th level feature map and l is the number of feature levels. Then, the features $\mathbf{x}$ with some learnable object queries are sequentially passed through a decoder containing T independent decoder layers $\{f_1, f_2, ..., f_T\}$. The specific decoding process can be formulated as follows:

$$\mathbf{y}_t = f_t(\mathbf{x}, \mathbf{y}_{t-1}|\theta_t), \tag{1}$$

where $\mathbf{y}_t$ denotes the queries (or termed as the latent variables) at step t outputted by layer f_t, and θ_t is the corresponding parameters of layer f_t. The main differences among current detectors are the definition of their object query $\mathbf{y}$ and the design of decoder layer f. Next, we will give a brief introduction to the AdaMixer design, and the detailed structure of AdaMixer decoder layer is illustrated in Fig. 2.

Object query of AdaMixer. In the AdaMixer object detector, an object query is decomposed into two parts: a content query vector and a positional query vector:

$$\mathbf{y}_t = (\mathbf{p}_t, \mathbf{q}_t), \tag{2}$$

where $\mathbf{q}_t \in \mathbb{R}^D$ is the content vector of $\mathbf{y}_t$, and $\mathbf{p}_t$ is the corresponding positional vectors. The content vector is expected to encode the appearance of object instances. The positional vectors represent the coordinates of an individual bounding box. Specifically, the positional vector in [14] is parameterized as $(x, y, z, r) \in \mathbb{R}^4$, and its relation with the bounding box is as follows:

$$\begin{aligned} x = x_{box}\ , &\quad y = y_{box}\ , \\ z = \log_2(\sqrt{wh})\ , &\quad r = \log_2(\frac{h}{w})\ , \end{aligned} \tag{3}$$

where (x_{box}, y_{box}) denotes the coordinates of the center point of the bounding box, and w, h indicate the width and height of this box.

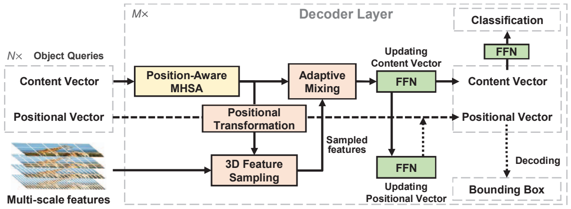


Figure 2: The detailed structure of AdaMixer [14] decoder layer. The object query is decoupled into a content vector and a positional vector. The decoder operates on these two types of vectors and refine them through a dynamic 3D feature sampling module and an adaptive mixing module.

Decoder layer of AdaMixer. As shown in Fig. 1a, the object queries are sequentially passed through the decoder layers to refine features and boxes. Each decoder layer of AdaMixer is typically composed of a multi-head self-attention module, a multi-head dynamic interaction module and some feed-forward networks (FFN), as illustrated in Fig. 2.

The object queries are first fed into the multi-head self-attention module, where the pairwise interaction is performed among queries. Then, the outputs, *i.e.*, the updated content vectors, are fed into the dynamic interaction module (3D feature sampling and adaptive mixing). In this module, a set of image features are first sampled from the extracted multi-scale features according to the object queries, and then the adaptive mixing are performed on these features. The processed features are then added into the content vectors.

Subsequently, each updated vector is fed into FFNs to predict the relative scaling and offsets to the positional vector (for generating a new bounding box) and the classification scores. Finally, the updated positional vectors (bounding boxes) and content vectors are sent into the next decoder layer.

3.2. DEQDet

After introducing the AdaMixer detector from a FNN view, we are ready to propose our DEQDet to improve it from both aspects of parameter efficiency and modeling capacity. We first reformulate AdaMixer from a RNN perspective to improve its parameter efficiency, and then further extend the modeling capacity of RNNDet to infinite refinement with deep equilibrium decoder.

Decoder from FFN to RNN. As depicted in Fig. 1a and stated in Eq. (1), the original query-based object detector works in a feed-forward way, where different decoder layers do not share weights and the resulting query vectors have different feature spaces. We argue this mechanism leads to large numbers of parameters and might be prone to overfitting. Instead, we observe that each decoder layer shares the same architecture and performs the progressive refinement of query vectors. In this view, RNN might be a more parameter-efficient solution by incorporating weight-tying mechanism among different refinement layers, as shown in Fig. 1b. In practice, this weight-tying strategy turns out not only the parameter is efficient, but also the detection performance can be improved.

Specifically, RNN iteratively processes the inputs with the same transformation and parameters. Formally, given a data sequence $[\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T]$ over time length (refinement step) T, the RNN decoder layers (except for initialization layer) share the same basic mathematical representation:

$$\mathbf{y}_t = f(\mathbf{x}_t, \mathbf{y}_{t-1}|\theta), \tag{4}$$

where θ is the parameters of the RNN function f, $\mathbf{y}_t$ is the latent variable produced by the function f at the time step t, and $\mathbf{y}_{t-1}$ is the preceding latent variable at the time step $t-1$. Actually, the RNNDet processes a special data sequence, where every data item $\mathbf{x}_i$ is set to be the same multi-scale features $\mathbf{x}$ and $\mathbf{y}_t$ represents the object queries. RNN is typically optimized through the BPTT [41]. The gradient flow of BPTT is illustrated in Fig. 3a.

Decoder from RNN to DEQ. Since RNN performs identical transformation on the inputs, the number of the iterations in Eq. (4) can be easily extended to the infinity if all the $\mathbf{x}_t$ share the same value. Furthermore, according to [3], when the sufficient stability condition is satisfied, the outputs of the weight-sharing layers of a general neural network tend to converge to a stable state as the model depth increases to infinity. In other words, when $t \to \infty$, the refinement layer would bring "diminishing return" and the network reaches an equilibrium:

$$\lim_{t\to\infty} \mathbf{y}_t = \lim_{t\to\infty} f(\mathbf{x}, \mathbf{y}_t|\theta) \triangleq \mathbf{y}^*, \tag{5}$$

where $\mathbf{y}^*$ indicates the fixed point (or called the equilibrium representation, the infinite feature representation). We can directly solve this fixed point as a root-finding problem [3]:

$$\mathbf{y}^* = f(\mathbf{x}, \mathbf{y}^*|\theta). \tag{6}$$

With this formulation, we can perform analytical backward pass in a constant memory consumption without tracing through the forward root-finding process.

The deep equilibrium decoder. To scale up detector into infinite-level refinement, we build our DEQDet based on the fixed point of an implicit layer. The overview of DEQDet architecture is presented in Fig. 1c. In our framework, there are only two types of layers in our decoder: an *initialization layer* and a *refinement layer*. The initialization layer first takes object queries as the input, and generates the image-related content vectors with image features and the coarse bounding box predictions:

$$\mathbf{y}_0 = g(\mathbf{x}, \mathbf{y}_{0^-}|\eta), \tag{7}$$

where $\mathbf{x}$ denotes the multi-scale image features extracted from the backbone (*e.g.*, like a conventional neural network [18, 42] or a vision transformer [28, 17]), the function g refers to the initialization layer with η as parameters, $\mathbf{y}_{0^-}$ denotes the initial object queries, and $\mathbf{y}_0$ denotes the object queries after initialization layer. The refinement layer is an implicit layer which models the infinite refinement, as define in Eq. (6), and its output is the fixed-point of this implicit layer. To solve the value of $\mathbf{y}^*$, we can resort to naive solver or quasi-Newton methods (*e.g.*, Anderson mixing [1]), and set $\mathbf{y}_0$ as the initial value in these methods.

3.3. Training of DEQDet

We first introduce the gradients of an implicit layer and present a tractable approximation method which is widely used in previous works. Then, we propose our *refinement aware gradient* (RAG) and *refinement aware perturbation* (RAP) for the effective training of our DEQDet.

Gradients of an implicit layer. To differentiate through the implicit layer defined by Eq. (6), the gradient of θ and $\mathbf{x}$ under $\mathbf{y}^*$ can be derived from Implicit Function Theorem (IFT) as follows:

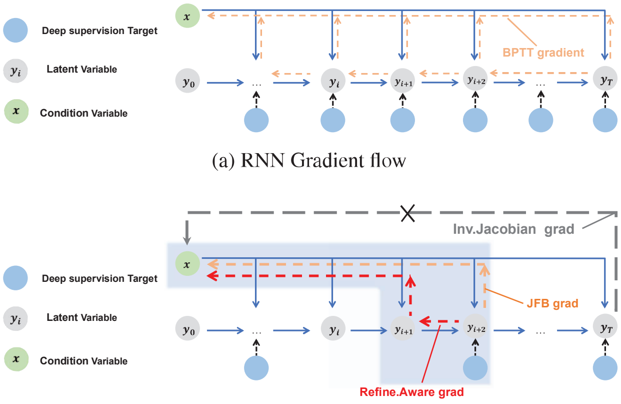
$$\frac{\partial \mathbf{y}^*}{\partial(\cdot)} = \left[I - \frac{\partial f(\mathbf{x}, \mathbf{y}^*|\theta)}{\partial \mathbf{y}^*}\right]^{-1} \frac{\partial f(\mathbf{x}, \mathbf{y}^*|\theta)}{\partial(\cdot)}, \tag{8}$$

where the variables in $(\cdot)$ can be $\mathbf{x}$ or θ, and the inverse-jacobian term $\left[I - \frac{\partial f(\mathbf{x},\mathbf{y}^*|\theta)}{\partial \mathbf{y}^*}\right]^{-1}$ is the most intriguing part in gradient solving. The original DEQ model integrates this term with VJP automation differential mechanism. VJP transforms the gradient solving to another linear *fixed-point system*, and thus it can also be solved via a *fixed-point solver* off the shelf [3]. However, the fixed-point iteration for the gradient solving requires huge computational consumption, thereby prohibiting the application for real scenarios [15].

Approximation of the inverse-Jacobian term. Following the recent works on backward gradient solving of implicit model [12, 2], we turn to estimate the inverse jacobian term because the resource consumption of the estimation method is relatively more acceptable. Specifically, Jacobian Free Backpropagation (JFB) [12, 2] approximates the gradient formula Eq. (9) by simply replacing the inverse jacobian term $\left[I - \frac{\partial f(x,y^*)}{\partial y^*}\right]^{-1}$ in Eq. (8) with identity matrix I:

$$\frac{\partial \mathbf{y}^*}{\partial(\cdot)} = I \cdot \frac{\partial f(\mathbf{x}, \mathbf{y}^*|\theta)}{\partial(\cdot)}. \tag{9}$$

Although JFB avoids the overhead of inverse gradient calculations and achieves good results in some tasks [2], in fact such a simple estimation ignores the *refinement* property of the function. The JFB gradient does not capture the relationship between input query $\mathbf{y}$ and updated query $f(\mathbf{x}, \mathbf{y}|\theta)$.



(a) RNN Gradient flow

(b) DEQ Gradient flow

Figure 3: **Gradient flow of DEQ model and RNN model**. The blue lines indicate forward flow. The dashed lines indicate the gradient flow. We use sparse deep supervision to train our DEQDet.

Therefore, in certain tasks that require high-level semantic understanding or have the sparsity, such as object detection, adopting JFB is not satisfactory.

Refinement-aware gradient and deep supervision. To capture the *refinement* nature of the decoder layer, we extend the Eq. (6) to a two-step unrolled equilibrium equation:

$$\mathbf{y}^* = f(\mathbf{x}, f(\mathbf{x}, \mathbf{y}^*|\theta)|\theta). \tag{10}$$

Based on Eq. (10), we propose our refinement aware gradient (detailed derivation in Appendix):

$$\frac{\partial \mathbf{y}^*}{\partial(\cdot)} \approx \left[I + \frac{\partial f(\mathbf{x}, \mathbf{y}^*|\theta)}{\partial \mathbf{y}^*}\right] \frac{\partial f(\mathbf{x}, \mathbf{y}^*|\theta)}{\partial(\cdot)}. \tag{11}$$

Note that Eq. (11) is also equivalent to 2-step Neumann-series-based Phantom Gradient [15]. Exactly, $\frac{\partial f(\mathbf{x},\mathbf{y}^*|\theta)}{\partial \mathbf{y}^*}$ is the refinement gradient term. Thus, in the Neumann-series of the inverse Jacobian term, its term $\sum_{i=1}^{k}[\frac{\partial f(\mathbf{x},\mathbf{y}^*|\theta)}{\partial \mathbf{y}^*}]^i$ controls the refinement awareness. We illustrate our refinement aware gradient with others in Fig. 3b.

As DEQ-Flow [2] suggests, employing sparse deep supervision on the fixed point solving path can improve the model performance. From the perspective of optimal transport, the refinement layer tries its best to transfer the input queries $\mathbf{y_t}$ to our desired queries $\hat{\mathbf{y}}$ whose decoding boxes are identical to the true distribution in the given image. Thus at each refinement step t, the refinement layer will deliver the most closet value $\mathbf{y}_{t+1}$ to the desired queries $\hat{\mathbf{y}}$. This suggests the smaller t, the greater difficulty, Therefore, we choose to construct supervision positions set Ω in following way:

$$\Omega_{\text{multiple}} = \{1, C, 2C, ..., m * C, T\}, \tag{12}$$

where m, C are constant numbers.

Refinement-aware perturbation. To further enhance the refinement awareness and improve the robustness of DEQDet, we introduce refinement-aware perturbation.

A general way of adding Gaussian noise. A simple noise-based perturbation is achieved by adding random noise to the latent variable $\mathbf{y}$, allowing the networks to recover from the corrupt result:

$$\hat{\mathbf{y}}_n = \mathbf{y}_n + \epsilon, \;\; \epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I) \tag{13}$$

where $\mathcal{N}(\cdot,\cdot)$ denotes a Gaussian distribution, σ denotes the noise scale, and ϵ is the random variable sampled from this distribution. However, directly adding this random noise is hard to simulate the real noise in fixed point solving process. To tackle this, we introduce a new refinement-aware perturbation approach. We propose to use the refinement Jacobian matrix $\frac{\partial \mathbf{y}_n}{\partial \mathbf{y}_{n-1}}$ to project a random noise to the latent space:

$$\hat{\mathbf{y}}_n = \mathbf{y}_n + \frac{\partial \mathbf{y}_n}{\partial \mathbf{y}_{n-1}} \cdot \epsilon, \;\; \epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I), \tag{14}$$

This approach can also be extended to a multi-step refinement-aware perturbation:

$$\hat{\mathbf{y}}_n = \mathbf{y}_n + \sum_{m=0}^{n-1} \mathbb{1}_{m \in \Psi} \cdot \frac{\partial \mathbf{y}_n}{\partial \mathbf{y}_m} \cdot \epsilon, \;\; \epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I), \tag{15}$$

where Ψ is the perturbation position set, indicating the indices of the solving path added with noise. This set is generated by probabilistic sampling, like random masking.

Adding noise to object detector. As for the specific implementation of adding noise to our detector, we treat the content vector $\mathbf{q}$ and positional vector $\mathbf{p}$ in different ways, as their physical meaning is not identical. For the positional vector, we first decode it to the corner-format bounding box (top-left point and bottom-right point), and then we add noise to these two points. Note that this noise-adding operation may cause the flip between these two corner points. Last, we transform the noise boxes to noise positional vectors. As for the content vector, we construct Gaussian noise with variance $\|\mathbf{q}\|_2^2$, linearly mixing the noise and content vector with the perturbation size σ_q:

$$\hat{\mathbf{q}} = (1-\sigma_q)\mathbf{q} + \sigma_q \epsilon, \;\; \epsilon \sim \mathcal{N}(\mathbf{0}, \|\mathbf{q}\|_2^2 I). \tag{16}$$

To impose the refinement Jacobian matrix on the noise term, in practice, we choose to directly feed the noisy latent variables into the refinement layer. Then, the gradients provided by the noise term is equivalent to have the refinement Jacobian matrix as the multiplier. The detailed demonstration and the noise perturbation algorithm can be found in Appendix.

4. Experiments

We conduct experiments on the MS-COCO 2017 dataset [25]. The training batch size is set to 16. We employ AdamW optimizer [29] to update the parameters with weight decay 0.01 for backbone and weight decay 0.1 for decoder, The loss consists of focal loss [24] with loss weight $\lambda_{\text{focal}} = 5$, L1 loss with loss weight $\lambda_{\text{L1}} = 5$ and GIoU loss [34] with loss weight $\lambda_{\text{giou}} = 2$. The matching cost for label assignment is aligned with loss. By default, we use the *fixed-point* iteration steps $T_{\text{train}} = 20$ for training and $T_{\text{infer}} = 25$ for inference as there is just little performance gain in further increasing T_{infer} . We place the detailed refinement steps experiment in Appendix. The base learning rate during training is 2.5×10^{-5}, and the lr multiplier for decoder is 4, we report the *mAP* performance on COCO *minival* set [25].

Detectors	AP	AP$_{50}$	AP$_{75}$	AP$_s$	AP$_m$	AP$_l$
FCOS [38]	38.7	57.4	41.8	22.9	42.5	50.1
Cascade R-CNN [5]	40.4	58.9	44.1	22.8	43.7	54.0
GFocalV2 [22]	41.1	58.8	44.9	23.5	44.9	53.3
BorderDet [32]	41.4	59.4	44.5	23.6	45.1	54.6
Dynamic Head [10]	42.6	60.1	46.4	26.1	46.8	56.0
DETR [6]	20.0	36.2	19.3	6.0	20.5	32.2
Deform-DETR [46]	35.1	53.6	37.7	18.2	38.5	48.7
Sparse R-CNN [35]	37.9	56.0	40.5	20.7	40.0	53.5
AdaMixer$_{T=6}$[14]	42.7	61.5	45.9	24.7	45.4	59.2
AdaMixer$^{\dagger}_{T=6}$[14]	42.7	61.5	46.1	24.9	45.5	59.3
RNNDet$_{T=6}$	43.4	62.0	46.5	26.3	46.1	58.8
RNNDet$_{T=12}$	44.2	63.1	47.7	26.1	47.0	60.0
DEQDet	**45.3**	**64.0**	**48.9**	**27.7**	**47.9**	**61.5**
DEQDet†	**46.0**	**64.8**	**49.6**	**27.5**	**49.0**	**61.4**

Table 1: **Comparison with other detectors under classic 1× training scheme with 100 queries**. RNNDet and DEQDet consist of Initialization layer and Refinement layer and trained with deep supervision. † means all layers with 64 sampling points instead of 32 sampling points

4.1. Classic 1× Training Results

We first report the performance of DEQDet by adopting the classic 1× training scheme. The classic 1× training scheme contains 12 training epochs with training images of shorter side resized to 800 and only with random flip data augmentation. In this study, the object query number is set to 100. We present the detailed results of 1× training results of RNNDet and DEQDet in Tab. 2 and compare with other detectors in Tab. 1. First, we compare the results between AdaMixer and RNNDet. With the same number of refinement (NF=6), RNNDet obtains the better performance than AdaMixer (43.4 vs. 42.7) with less than half parameters and similar inference speed. This superior performance verifies the effectiveness of weight-tying strategy. Second, we increase the refinement number in RNNDet from 6 to 20, and obtain the best performance of 44.2 at NF= 8 or 12. However, we can clearly observe that as the number of refinement layers in RNNDet further increases, the performance degrades partially due to RNN optimization difficulty. We visualize the gradient norm in Fig. 4 and the RNN norm is not stable.

Detectors	NF	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	Params	FPS	$\text{Mem}_{\text{Train}}$	$\text{Mem}_{\text{Infer}}$	TrainTime
AdaMixer[14]	6	42.7	61.5	45.9	24.7	45.4	59.2	134M	13.5	5961M	872M	$\sim$ 14.0h
AdaMixer†[14]	6	42.7	61.5	46.1	24.9	45.5	59.3	160M	13.5	6803M	972M	$\sim$ 15.5h
RNNDet	6	43.4	62.0	46.5	26.3	46.1	58.8	61M	13.6	4517M	588M	$\sim$ 12.0h
	8	44.2	62.9	47.9	26.5	47.2	60.1	61M	12.0	4784M	588M	$\sim$ 13.5h
	12	44.2	63.1	47.7	26.1	47.0	60.0	61M	9.2	5567M	588M	$\sim$ 17.5h
	16	43.8	62.7	47.2	26.9	46.7	59.1	61M	7.7	6195M	588M	$\sim$ 22.0h
	20	43.7	62.5	47.0	26.5	46.4	59.6	61M	5.9	6818M	588M	$\sim$ 24.0h
DEQDet	6	44.3	63.0	47.6	26.2	47.1	60.5	61M	13.6	4827M	588M	$\sim$ 25.5h
	8	44.9	63.6	48.2	27.0	47.5	61.1	61M	12.0			
	16	45.2	63.9	48.8	27.5	47.9	61.3	61M	7.7			
	25	**45.3**	**64.0**	**48.9**	**27.7**	**47.9**	**61.5**	61M	5.1			
DEQDet†	6	45.7	64.3	49.3	27.5	48.7	61.9	69M	13.0	4997M	622M	$\sim$ 29.0h
	25	**46.0**	**64.8**	**49.6**	**27.5**	**49.0**	**61.4**	69M	4.8			

Table 2: **classic 1× training results** with 100 queries. RNNDet and DEQDet consist of Initialization layer and Refinement layer and trained with deep supervision. † means all layers with 64 sampling points instead of 32 sampling points

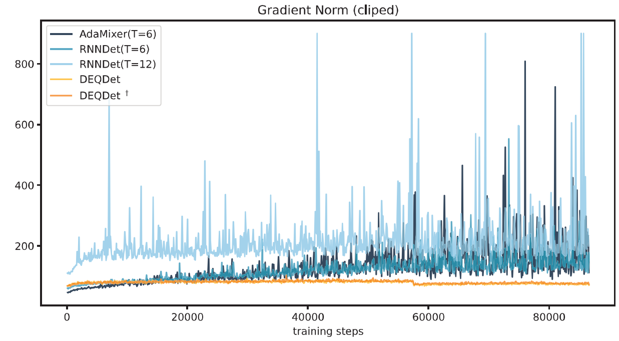


Figure 4: **Gradient Norm of Detectors**, To obtain the gradient norm , we first flatten all gradients as a single vector, then calculate the l2-norm of this gradient vector.

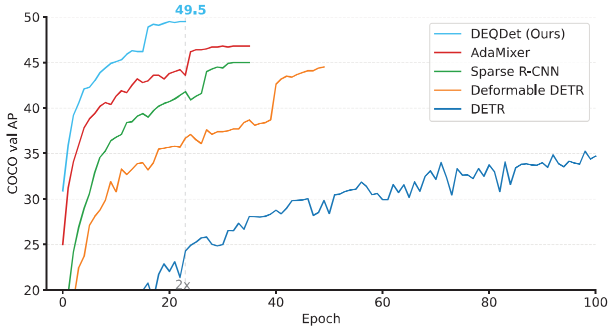


Figure 5: **Trainng Convergence Curves** of DEQDet and AdaMixer [14], Sparse-RCNN [35], DETR [6], Deformable DETR [46]. The number of object queries is 300 and backbone is ResNet50.

Then, we present the result of our DEQDet and see the gradient norm of DEQDet is very consistent and stable. When NF in the fixed-point solving process is set to 6, our DEQDet achieves better performance (44.3) with less parameters and smaller memory consumption than AdaMixer. When we further increase the NF in DEQDet to 25, it obtains the best performance of 45.3 under 32 sampling points and 46.0 under 64 sampling points. Finally, we notice that the inference time of DEQDet is comparable to the other counterparts, but its training time is relatively larger due to the forward fixed-point solving process.

We also compare our DEQDet with other detectors under this limited training epochs and data augmentations in Tab. 1. The results demonstrate that our DEQDet achieves significant improvement over previous detectors under this limited training budget. This result show that our DEQDet is training-efficient and provides a highly competitive baseline for future object detector design.

4.2. Comparison with the state of the art

After reporting the results under limited training budget, we will scale up our DEQDet with more object queries, longer training epochs, and stronger backbones. First, we visualize the training convergence curves of R50 backbone and 300 queries in Fig. 5. DEQDet is designed to explore the potential power of refinement layer as much as possible. Compared with our counterpart AdaMixer [14], DEQDet convergences faster and achieves higher *mAP*. We conjecture the gradient norm in DEQDet is consistent and stable as illustrated in Fig. 4, which leads to faster convergence.

We compare the results of our DEQDet with other detectors in Tab. 3. In Tab. 3, we allocate 300 queries in DEQDet Our DEQDet shows fast convergence and thus we only scale the training epochs to 24, which is smaller than previous methods. We observe that under the backbone of ResNet50, our DEQDet† achieves 49.5 *mAP* under 2× training scheme.

Detector	Backbone	Encoder/FPN	Epochs	Params	AP	AP$_{50}$	AP$_{75}$	AP$_s$	AP$_m$	AP$_l$
DETR [6]	ResNet-50-DC5	TransformerEnc	500	41M	43.3	63.1	45.9	22.5	47.3	61.1
SMCA [13]	ResNet-50	TransformerEnc	50	40M	43.7	63.6	47.2	24.2	47.0	60.4
Deformable DETR [46]	ResNet-50	DeformTransEnc	50	40M	43.8	62.6	47.7	26.4	47.1	58.0
Anchor DETR [40]	ResNet-50-DC5	DecoupTransEnc	50	35M	44.2	64.7	47.5	24.7	48.2	60.6
Efficient DETR [43]	ResNet-50	DeformTransEnc	36	35M	45.1	63.1	49.1	28.3	48.4	59.0
Conditional DETR [30]	ResNet-50-DC5	TransformerEnc	108	44M	45.1	65.4	48.5	25.3	49.0	62.2
Sparse R-CNN [35]	ResNet-50	FPN	36	110M	45.0	63.4	48.2	26.9	47.2	59.5
REGO [9]	ResNet-50	DeformTransEnc	50	54M	47.6	66.8	51.6	29.6	50.6	62.3
DAB-D-DETR [26]	ResNet-50	DeformTransEnc	50	48M	46.8	66.0	50.4	29.1	49.8	62.3
DN-DAB-D-DETR [21]	ResNet-50	DeformTransEnc	12	48M	43.4	61.9	47.2	24.8	46.8	59.4
DN-DAB-D-DETR [21]	ResNet-50	DeformTransEnc	50	48M	48.6	67.4	52.7	31.0	52.0	63.7
AdaMixer [14]	ResNet-50	-	12	139M	44.1	63.1	47.8	29.5	47.0	58.8
AdaMixer [14]	ResNet-50	-	24	139M	46.7	65.9	50.5	29.7	49.7	61.5
AdaMixer [14]	ResNet-50	-	36	139M	47.0	66.0	51.1	30.1	50.2	61.8
RNNDet$_{T=8}$	ResNet-50	-	36	65M	48.1	66.7	52.3	31.2	51.1	62.5
RNNDet$^{\dagger}_{T=8}$	ResNet-50	-	36	69M	48.4	67.1	52.7	31.8	51.4	63.4
DEQDet	ResNet-50	-	12	65M	46.6	65.3	50.6	30.5	49.4	61.2
DEQDet	ResNet-50	-	24	65M	**48.6**	**67.6**	**53.0**	**31.6**	**51.8**	**62.9**
DEQDet†	ResNet-50	-	24	69M	**49.5**	**68.1**	**53.9**	**33.0**	**52.0**	**63.3**
DETR [6]	ResNet-101-DC5	TransformerEnc	500	60M	44.9	64.7	47.7	23.7	49.5	62.3
SMCA [13]	ResNet-101	TransformerEnc	50	58M	44.4	65.2	48.0	24.3	48.5	61.0
Efficient DETR [43]	ResNet-101	DeformTransEnc	36	54M	45.7	64.1	49.5	28.2	49.1	60.2
Conditional DETR [30]	ResNet-101-DC5	TransformerEnc	108	63M	45.9	66.8	49.5	27.2	50.3	63.3
Sparse R-CNN [35]	ResNet-101	FPN	36	125M	46.4	64.6	49.5	28.3	48.3	61.6
REGO [9]	ResNet-101	DeformTransEnc	50	73M	48.5	67.0	52.4	29.5	52.0	64.4
AdaMixer [14]	ResNet-101	-	36	158M	48.0	67.0	52.4	30.0	51.2	63.7
DEQDet	ResNet-101	-	24	84M	**49.5**	**68.2**	**53.8**	**33.6**	**52.8**	**64.3**
DEQDet†	ResNet-101	-	24	88M	**50.1**	**68.9**	**54.5**	**34.3**	**53.3**	**65.1**
AdaMixer [14]	Swin-S	-	36	164M	51.3	71.2	55.7	34.2	54.6	67.3
DEQDet	Swin-S	-	24	90M	**52.7**	**72.3**	**57.6**	**36.6**	**55.9**	**68.4**

Table 3: Comparison with other detectors on COCO *minival* set. The number of queries defaults to 300 in our DEQDet. † means refinement layer with 64 sampling points.

outperforming its baseline Adamixer by 2.5 *mAP*. Especially in small object detection metrics, DEQDet† achieves 33.0 AP_s. We also scale the backbone of our DEQDet to ResNet-101 and Swin-S. Our DEQDet can outperform the AdaMixer by 2.1 *mAP* for ResNet101 and 1.4 *mAP* for Swin-S. This shows our DEQDet generalizes well to large backbones.

4.3. Ablation studies

In this ablation study, we use 100 queries and ResNet50 as the backbone for DEQDet. The training epoch is 12.

Initialization layer. As introduced in Sec. 3.2, we employ an initialization layer to convert the image content agnostic queries to image content-related queries. We investigate different initialization layer setting in Tab. 4a, including 1). no initialization layer, 2). an initialization layer with 32 sampling points 3). an initialization layer with 64 sampling points. As initialization layer with enough sampling points can obtain relatively rich semantic information from input features, it will releive the learning difficulty of subsequent layers. Another question is whether to apply extra supervision to initialization layer during training. We summarize $h = 0, 1, 2$ in Tab. 4b, where h represents the number of extra layers placed on top of initialization layer for supervision. When $h = 0$, there is no connection between initialization layer and refinement layer, so the training is unstable.

Refinement-aware gradient. As discussed in Sec. 3.3, due to the high-level semantic understanding property and sparse nature of object detection, ignoring the refinement aware gradient will lead to sub-optimal results. We verify our conjecture through experiments in the Tab. 4c. The refine-

Init layer sampl.points	AP	AP_{50}	AP_{75}
/	45.2	63.8	48.8
32	45.3	64.0	48.9
64	45.5	64.4	49.1

(a) **Init Layer**. A large Init layer benifits performance.

extra super. layers	AP	AP_{50}	AP_{75}
0	44.7	63.4	48.2
1	45.1	63.8	48.8
2	45.5	64.4	49.1

(b) **Init Layer supervised** with extra refinement aware gradient.

RAG step.k	AP	AP_{50}	AP_{75}
1	41.9	60.7	45.0
2	45.5	64.4	49.1
3	45.5	64.2	49.4
4	45.7	64.2	49.9

(c) **refinement aware gradient** with different k.

m	C	AP	AP_{50}	AP_{75}
4	3	45.5	64.4	49.1
4	4	45.4	63.9	49.2
3	3	45.1	64.0	48.9
5	3	45.2	63.8	48.8

(d) **deep supervision position set Ω.**

perturbation step.	AP	AP_{50}	AP_{75}
zero-step	44.9	63.8	48.6
one-step	45.1	63.9	48.6
multi-step	45.5	64.4	49.1

(e) **perturbation step** multiple step perturbation works best.

σ_q	σ_p	AP	AP_{50}	AP_{75}
/	/	44.3	63.1	47.8
/	25	45.4	64.2	49.0
/	50	45.5	64.5	49.1
0.1	/	45.0	63.6	48.5
0.2	/	45.2	64.3	48.6

(f) **single perturbation noise.**

σ_q	σ_p	AP	AP_{50}	AP_{75}
0.1	25	45.5	64.4	49.1
0.1	50	45.5	64.4	49.1
0.2	25	45.2	63.9	49.0
0.2	50	45.2	63.9	49.2

(g) **perturbation noise combinations.**

iteration steps	AP	AP_{50}	AP_{75}
15	45.1	63.8	48.7
20	45.5	64.4	49.1
25	45.3	64.0	48.9

(h) ***fixed-point* iteration steps** during training.

Table 4: **Ablation Studies** on our DEQDet design with ResNet-50 as the backbone and 100 object queries under the 12 training epochs on the MS-COCO *minival* set.

ment aware gradient step k refers to the expansion steps of neumann-series. When $k = 1$, RAG degenerates into JFB [12], which fails to consider the refinement property of *fixed-point* iteration. $k = 2$ is the standard RAG baseline, which achieves 45.5 mAp. RAG outperforms JFB substantially in object detection task. Although the $k = 3$ setting retains the locality property and enjoys more refinement awareness, it offers little improvement. So, we set $k = 2$.

Deep supervision position Ω. We experiment with position set in Eq. (12) under different settings. We use the default RAG step $k = 2$. We experimentally find using Ω_{multiple} with $m = 4, C = 3$ achieves the best result.

Refinement-aware perturbation. We compare zero-step simple noise and perturbation noise in Tab. 4e. As introduced in Sec. 3.3, one-step perturbation noise naively projects the simple noise by refinement layer to latent space. One-step noise improves the simple noise by 0.2 mAP, which means adding the noise associated with fixed-point solving is better than simple Gaussian noise. The best results are achieved with multi-step noise, as it takes full advantage of the fixed-point solution path.

Perturbation noise. In Tab. 4f, we experiment with different noise scales in terms of position noise perturbation and content perturbation and their combinations. From Tab. 4f, we find both content perturbation and position perturbation significantly improve the performance of DEQDet. The improvement effect of position perturbation is more obvious than that of content perturbation. As the noise scale increases, the performance can be further improved. However, Tab. 4g, when we combine the best content noise $\sigma_q = 0.2$ and the best best position noise $\sigma_p = 50$, DEQDet yields only 45.2 *mAP*. The Performance degradation of perturbation combination shows excessive noise perturbation hurts DEQDet performance. An reasonable noise scale $\sigma_q = 0.1$ and $\sigma_p = 25$ achieves the best *mAP* (45.5).

Refinement iteration steps. We also experiment with different *fixed-point* iteration steps T_{train} during training of DEQDet. Tab. 4h shows $T_{\text{train}} = 20$ achieves the highest mAP performance(45.5). Comparing other steps, we can conclude that more refinement steps during training can enhance the performance of detectors. But $T_{\text{train}} = 25$ does not exceed $T_{\text{train}} = 20$, which may be due to limited deep supervision positions for those redundant refinements.

5. Conclusion

In this paper, we have proposed the deep equilibrium detector (DEQDet), a query-based object detector with infinite refinement steps. We equivalently model the refinement process as a *fixed-point* solving problem of a implicit layer. As for the training of DEQDet, we find that its simple estimation on inverse-jacobian term lacks refinement awareness, resulting in a negative impact on the high-level semantic understanding of the object detector. Therefore, to inject the refinement awareness into the detector during training, we propose the refinement-aware gradient (RAG) and the refinement-aware perturbation (RAP). Our experiments show DEQDet converges faster, consumes less memory, and achieves better performance than the counterparts on the MS-COCO dataset. We hope our DEQDet becomes a strong baseline and could inspire future work to consider deep equilibrium modeling in other computer vision tasks.

Acknowledgements. This work is supported by National Key R&D Program of China (No. 2022ZD0160900), National Natural Science Foundation of China (No. 62076119, No. 61921006).

References

[1] Donald G Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965. 5

[2] Shaojie Bai, Zhengyang Geng, Yash Savani, and J Zico Kolter. Deep equilibrium optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 620–630, 2022. 3, 5

[3] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 3, 4, 5

[4] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *Advances in Neural Information Processing Systems*, 33:5238–5250, 2020. 3

[5] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: high quality object detection and instance segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1483–1498, 2019. 1, 3, 6

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European conference on computer vision*, pages 213–229, 2020. 1, 2, 3, 6, 7, 8

[7] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018. 3

[8] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. Diffusiondet: Diffusion model for object detection. *arXiv preprint arXiv:2211.09788*, 2022. 3

[9] Zhe Chen, Jing Zhang, and Dacheng Tao. Recurrent glimpse-based decoder for detection with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5260–5269, 2022. 8

[10] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7373–7382, 2021. 6

[11] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019. 1

[12] Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley Osher, and Wotao Yin. Jfb: Jacobian-free backpropagation for implicit networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 2, 3, 5, 9

[13] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3621–3630, 2021. 3, 8

[14] Ziteng Gao, Limin Wang, Bing Han, and Sheng Guo. Adamixer: A fast-converging query-based object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5364–5373, 2022. 1, 2, 3, 4, 6, 7, 8

[15] Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. *Advances in Neural Information Processing Systems*, 34:24247–24260, 2021. 2, 5

[16] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 3

[17] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022. 5

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 3

[20] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision*, pages 734–750, 2018. 1

[21] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13619–13627, 2022. 8

[22] Xiang Li, Wenhai Wang, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11632–11641, 2021. 6

[23] Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, KiJung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning*, pages 3082–3091. PMLR, 2018. 3

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 6

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[26] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr. *arXiv preprint arXiv:2201.12329*, 2022. 3, 8

[27] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In

Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 10012–10022, 2021. 5

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[30] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3651–3660, 2021. 8

[31] Fernando Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *Neural information processing systems*, 1987. 3

[32] Han Qiu, Yuchen Ma, Zeming Li, Songtao Liu, and Jian Sun. Borderdet: Border feature for dense object detection. In *European Conference on Computer Vision*, pages 549–564. Springer, 2020. 6

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1, 3

[34] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019. 6

[35] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14454–14463, 2021. 1, 2, 3, 6, 7, 8

[36] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 2

[37] Yao Teng, Haisong Liu, Sheng Guo, and Limin Wang. StageInteractor: Query-based object detector with cross-stage interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 3

[38] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 1, 6

[39] Tiancai Wang, Xiangyu Zhang, and Jian Sun. Implicit feature pyramid network for object detection. *arXiv preprint arXiv:2012.13563*, 2020. 3

[40] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. In *Proceedings of the AAAI conference on artificial intelligence*, pages 2567–2575, 2022. 8

[41] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. 2, 4

[42] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 5

[43] Zhuyu Yao, Jiangbo Ai, Boxun Li, and Chi Zhang. Efficient detr: improving end-to-end object detector with dense prior. *arXiv preprint arXiv:2104.01318*, 2021. 8

[44] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Probabilistic two-stage detection. *arXiv preprint arXiv:2103.07461*, 2021. 1

[45] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 1

[46] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 3, 6, 7, 8