# iDETR: Implicit DETR for Tiny Object Detection

**Lingcong Cai**[1,*] **Haiqin Yang**[1,2,*] **Jikui Liu**[3]
**Yunpeng Cai**[4] **Yumeng Liu**[1,†] **Xiaomao Fan**[1,†]

[1] Shenzhen Technology University
[2] Institute of Global Sci-Tech Strategy and Security Studies, IGSSS-SZTU
[3] Shenzhen Polytechnic University
[4] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences

[*] Equal contribution. [†] Corresponding authors.

## Abstract

Tiny object detection (TOD) remains challenging due to the extremely limited spatial details of small objects. Existing methods heavily rely on multi-scale feature pyramids with discrete feature maps, which suffer from insufficient spatial resolution, quantization errors, and poor localization accuracy, while imposing heavy computational and memory overhead. We present the **i**mplicit **DE**tection **TR**ansformer (**iDETR**), an efficient detector that operates exclusively on low-resolution single-scale features. It incorporates two novel components: (1) **implicit Attention (iAttn)**, which leverages implicit neural representations to model continuous features and enables precise sub-pixel querying beyond discrete grid limitations; and (2) **Centroid-Guided Query Initialization (CGQI)** for robust query initialization under single-scale constraints. We further propose head-conditional sampling in iAttn, which reduces the querying computational cost by 4× and memory footprint by 3× without sacrificing performance. Extensive experiments demonstrate the effectiveness of iDETR. In particular, on AI-TODv2, iDETR outperforms state-of-the-art methods by 0.2% AP overall, with particularly strong gains of 1.1% AP on very tiny objects, while reducing computational cost by 37%. The code is available at `https://github.com/AnoK3111/iDETR`.

## 1 Introduction

Tiny object detection (TOD) focuses on accurately localizing and classifying objects with extremely limited pixel coverage, typically smaller than $16 \times 16$ pixels, as established in key TOD benchmarks [1, 2]. This task finds broad applications in real-world scenarios, including remote sensing, autonomous driving, and pedestrian detection. Although deep learning-based general object detectors have achieved remarkable progress [3, 4, 5, 6, 7, 8], TOD remains one of computer vision's most persistent challenges, primarily due to the inherent scarcity of discriminative features in tiny targets.

Multi-scale design has long been regarded as an effective strategy to enhance TOD performance. By preserving and fusing features across different resolutions from backbone stages, it supplies rich semantic and fine-grained spatial information tailored to objects of varying sizes. In CNN-based detectors, this divide-and-conquer paradigm–assigning feature maps of appropriate scales to objects of corresponding sizes–has proven highly successful [9, 10, 11]. For example, FCOS [12] saw its AP rise dramatically from 0.5 to 16.4 after adopting multi-scale design. More recently, DETR [13]

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

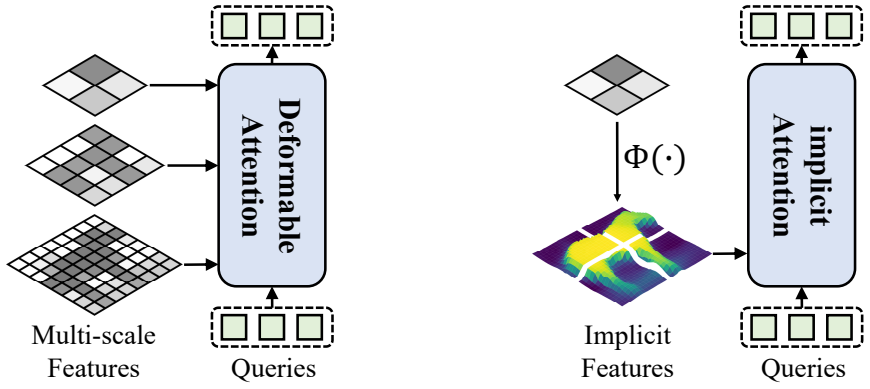



Figure 1: (a) DETR-based models employ deformable attention to query multi-scale discrete features. (b) Our iDETR leverages implicit attention to query continuous implicit features transformed from low-resolution single-scale features.

introduced an end-to-end paradigm that excels at global dependency modeling and eliminates post-processing such as NMS. To boost TOD performance and convergence in DETR-family models, multi-scale designs were reincorporated into leading variants such as Deformable DETR [14], DINO-DETR [4], and D-FINE [5]. These approaches generate higher-resolution features, enabling the decoder to access more precise spatial details critical for tiny objects.

Despite these advances, multi-scale designs inherently face two key limitations for TOD: (1) discrete feature maps provide limited fine-grained spatial information at inter-pixel positions, leading to quantization errors, aliasing, and poor localization of tiny objects that occupy only a few grid cells; and (2) fusing and processing high-resolution features incurs substantial computational and memory overhead, which escalates rapidly with input image resolution–a critical concern in high-resolution aerial imagery typical of TOD.

In this paper, we propose the **i**mplicit **DE**tection **TR**ansformer (**iDETR**), which discards multi-scale features and operates exclusively on low-resolution features, thereby substantially reducing computational burden. Our iDETR introduces two novel components: (1) **implicit Attention (iAttn)**, which converts discrete low-resolution features into continuous implicit representations, enabling fine-grained querying at arbitrary spatial positions and alleviating the resolution and quantization limitations of grid-based features; and (2) **Centroid-Guided Query Initialization (CGQI)**, which enhances query initialization under single-scale constraints by predicting candidate object centroids and dynamically aggregating high-quality features around them via iAttn. As illustrated in Fig. 1, iDETR replaces discrete multi-scale features with continuous implicit representations, providing richer spatial modeling while eliminating the heavy overhead of multi-scale design. Moreover, we introduce **head-conditional sampling** within iAttn, which integrates head-specific projections directly into the implicit sampler, reducing querying computational cost by $4\times$ and memory by $3\times$ without compromising performance.

In summary, our main contributions are three-fold: (1) We propose **iDETR**, the first TOD model to discard multi-scale and high-resolution features in favor of continuous implicit representations learned solely from low-resolution inputs. It introduces two core components: **iAttn** for transforming discrete low-resolution features into continuous implicit features for efficient, resolution-agnostic querying at arbitrary spatial positions, and **CGQI** for robust and accurate query initialization under single-scale constraints. (2) We further introduce **head-conditional sampling** within iAttn, which integrates head-specific projections directly into the implicit sampler, reducing the querying computational cost by $4\times$ and memory footprint by $3\times$ without sacrificing performance. (3) Extensive experiments show the effectiveness of iDETR. In particular, on AI-TODv2, iDETR outperforms state-of-the-art methods by 0.2% AP overall, with particularly strong gains of +1.1% AP on very tiny objects, while reducing computational cost by 37%.

## 2 Related Work

### 2.1 DETR-based Models

Object detection has long relied on convolutional neural networks (CNNs) with handcrafted components such as anchors and non-maximum suppression (NMS). However, the pioneering Detection Transformer (DETR) [13] introduced a transformative end-to-end framework that directly predicts object sets using transformers, eliminating the need for post-processing such as NMS via Hungarian matching for one-to-one label assignment. Despite its elegant simplicity and strong performance potential, DETR suffered from slow convergence and high computational demands, limiting its practical deployment.

Building on DETR, subsequent studies addressed these limitations. Deformable-DETR [14] incorporated multi-scale feature processing and a two-stage prediction scheme, where encoder output generated proposals to initialize decoder queries, improving efficiency and accuracy. To mitigate DETR's unstable matching and slow training, recent advances introduced denoising strategies [15, 4], one-to-many label assignment [16, 17], and dense one-to-one matching [6]. These innovations substantially accelerated convergence while boosting performance.

More recently, lightweight DETR variants emerged to enable real-time inference. RT-DETR [18] proposed a hybrid encoder that balances efficiency and accuracy by applying global attention to low-resolution features before fusing multi-scale information. D-FINE [5] further enhanced performance with minimal overhead by reformulating box regression as probability distribution refinement and incorporating self-distillation.

While DETR-based models advanced end-to-end detection and achieved competitive results, their heavy reliance on multi-scale features and high computational overhead hinders deployment in resource-constrained scenarios. This motivates our approach to discard explicit multi-scale processing and instead model continuous features, providing richer fine-grained representations with lower overhead.

### 2.2 Tiny Object Detection

Tiny object detection (TOD) remains a challenging task due to limited pixel coverage, occlusion, and background clutter. Existing TOD methods generally fall into four categories: multi-scale design, data augmentation, label assignment, and feature enhancement.

Multi-scale designs improve TOD by effectively fusing high-resolution (fine details) and low-resolution (semantic context) features [10, 19, 20, 3, 21]. Data augmentation generates more TOD samples; for example, copy-paste techniques [22] inserted additional tiny objects into images. Label assignment strategies, such as ATSS [23] and RFLA [24], provided more accurate supervision by refining positive/negative sample allocation. Feature enhancement methods further boost performance: SR-TOD [25] adopted self-reconstruction, SET [26] amplified high frequencies in the spectral domain, and entropy-minimization approaches [27] preserved information in feature maps. Despite these advances, TOD methods often require large input resolutions to preserve tiny object details, leading to substantial computational overhead and limiting real-time applicability.

### 2.3 Implicit Neural Representation

Implicit neural representations (INRs) learn continuous mappings from query coordinates to corresponding values (e.g., occupancy, color, or features). This paradigm excels in parameterizing signals at arbitrary resolutions without discretization artifacts and has gained widespread adoption in computer vision.

In 3D domains, INRs enable high-fidelity shape reconstruction [28, 29] and novel view synthesis [30, 31]. In 2D tasks, they support image super-resolution [32, 33], generation at arbitrary resolutions [34, 35], and other low-level vision problems by capturing continuous signals efficiently. The core advantage of INRs lies in their ability to represent dense, continuous spatial information, addressing the limitations of grid-based or discrete representations.

While INRs have proven highly effective for continuous signal modeling in 3D reconstruction and 2D image tasks, their application to object detection, particularly for providing fine-grained features

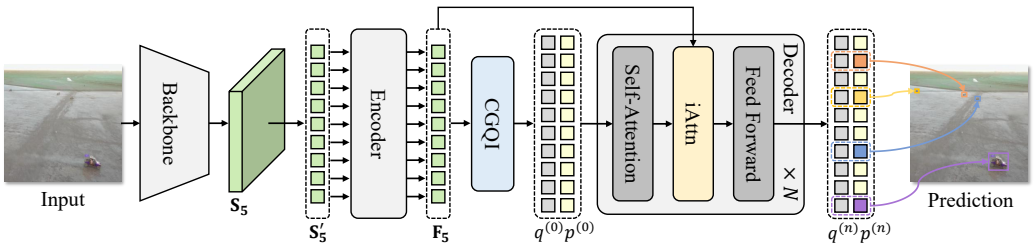



Figure 2: iDETR architecture overview. iDETR operates exclusively on single-scale low-resolution features $\mathbf{S}_5$, eliminating multi-scale feature pyramids. It introduces two core components: (1) **iAttn**, which replaces deformable attention with continuous sub-pixel feature querying via implicit neural representations; and (2) **CGQI**, which produces denser and more accurate initial queries by predicting object centroids and refining features around them with iAttn.

without multi-scale overhead—remains underexplored. Inspired by these successes, we leverage INR to model continuous features in detection pipelines, eliminating the need for expensive high-resolution discrete feature inference and enabling more efficient, high-performance models.

# 3 Methods

## 3.1 Deformable Attention in DETR-based Models

Modern DETR-based detectors heavily rely on deformable attention [5] in the decoder to iteratively refine object queries by aggregating relevant spatial information from feature maps. Given a feature map $F \in \mathbb{R}^{H\times W\times d}$ ($H$: spatial height, $W$: spatial width, and $d$: channel dimension), an object query $q_i \in \mathbb{R}^d$, and its corresponding 2D reference point $p_i \in [0,1]^2$ (normalized image coordinates), the deformable attention output for query $i$ is computed as:

$$\text{DeformAttn}(q_i, p_i, F) = \sum_{m=1}^{M} W_m \Big[ \sum_{k=1}^{K} A_{imk} \cdot W'_m f_s(F, p_i + \Delta p_{imk}) \Big], \tag{1}$$

where:

- $m$ indexes the attention head and $k$ indexes the $k$-th sampling point.
- $\Delta p_{imk} \in \mathbb{R}^2$ and $A_{imk} \in [0,1]$ are the sampling offset and attention weight for the $m$-th head and $k$-th point, both predicted from $q_i$ via a learnable linear projection.
- $W_m \in \mathbb{R}^{d\times d_h}$ and $W'_m \in \mathbb{R}^{d_h\times d}$ are learnable projection matrices for the $m$-th head ($d_h = d/M$ is the per-head dimension).
- $f_s(\cdot)$ is the feature sampling function (bilinear interpolation in standard implementations).

In the vanilla deformable attention, $f_s(\cdot)$ relies on bilinear interpolation over a discrete grid of feature values. This discretization inherently limits sampling precision, particularly when the reference point $p_i$ or predicted offsets fall between grid cells. As a result, fine-grained details—crucial for detecting tiny objects—are often lost or inaccurately represented unless very high-resolution feature maps are used, which significantly increases computational cost and memory consumption. This limitation becomes especially pronounced in TOD tasks, where objects occupy only a few pixels and require precise localization and feature extraction. The reliance on discrete, low- to mid-resolution grids hampers the model's ability to capture subtle spatial cues, leading to degraded performance on small-scale instances despite the efficiency advantages of deformable attention over standard dot-product attention.

## 3.2 iDETR: implicit Detection Transformer

Figure 2 illustrates the overall architecture of iDETR. Our core innovation is a continuous feature representation via implicit neural representations (INR), which overcomes the discretization limits of

standard feature maps. We model features as a continuous function $f_{\text{Inr}}(\cdot)$ learned from low-resolution inputs, enabling sub-pixel querying without high-resolution pyramids. We build on this with two key modules: (1) **iAttn** (see Sec. 3.2.1), which replaces discrete bilinear sampling in deformable attention with continuous sampling from $f_{\text{Inr}}(\cdot)$ for precise offset-based feature extraction. (2) **CGQI** (see Sec. 3.2.2), which predicts object centroids to initialize a dense, high-quality set of decoder queries.

### 3.2.1 iAttn: implicit Attention

Implicit Attention or iAttn encodes continuous features by replacing the original bilinear sampling function $f_s(\cdot)$ in deformable attention with a learnable INR-based sampling function $f_{\text{Inr}}(\cdot)$. Learning a single global INR function directly from low-resolution features can be challenging because it is difficult to capture diverse local patterns. To address this, we propose learning a family of local INR-based functions $\{\text{INRL}(\cdot;\theta_{i,j})\}$, each sharing the same network architecture but possessing distinct parameters $\theta_{i,j}$. These local functions are conditioned on and learned from individual latent codes extracted from the feature maps. Formally, let $z_{i,j} \in \mathbb{R}^d$ denote the latent code at spatial index $(i,j)$ of the feature map $F$; we define a parameter mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^p$ that maps $z_{i,j}$ to the parameters $\theta_{i,j}$ of its corresponding function, i.e. $\theta_{i,j} = \Phi(z_{i,j})$.

For any query point $p \in \mathbb{R}^2$, the fine-grained feature at $p$ is estimated by a learnable local INR-based function:

$$f_{\text{Inr}}(F,p) = \text{INRL}(p - p^*; \theta_{i^*,j^*}), \tag{2}$$

where $i^*$ and $j^*$ denote the indices of the latent code that is closest to $p$ under the Euclidean distance, and $p^*$ denotes the coordinates of $z_{i^*,j^*}$.

**Head-conditional sampling.** Vanilla deformable attention employs a two-stage sampling process (see Eq. (1)): it first samples features at the predicted offset using the sampling function $f_s(\cdot)$, then applies a head-specific linear projection $W'_m$ to reduce the sampled $d$-dimensional feature to $d/M$ per attention head. When integrating our local INR-based function $\text{INRL}(\cdot)$, this design becomes inefficient: $\text{INRL}(\cdot)$ must first generate a full $d$-dimensional output, only to be immediately projected down to $d/M$ via $W'_m$. This redundant high-dimensional computation introduces significant overhead. To eliminate this inefficiency, we fuse the head-specific dimensionality reduction directly into the local INR function by conditioning it on the attention head index $m$. The resulting unified sampling function is defined as:

$$f_{\text{Inrcs}}(F,p,m) = \text{INRLCS}(p - p^*, \gamma(m); \theta_{i^*,j^*}), \tag{3}$$

where $\text{INRLCS}(\cdot)$ denotes the conditional local INR-based function, and $\gamma(\cdot)$ represents the sinusoidal positional encoding function [36] that embeds the head index $m$ into a fixed-dimensional head embedding. This conditioning allows $\text{INRLCS}(\cdot)$ to directly output the head-specific $d/M$-dimensional feature, reducing the required output dimension by a factor of $M$ and eliminating the redundant matrix multiplication without any loss in performance. The modified deformable attention with implicit sampling becomes:

$$\text{iAttn}(q_i, p_i, F) = \sum_{m=1}^{M} W_m \Big[ \sum_{k=1}^{K} A_{imk} \cdot f_{\text{Inrcs}}(F, p_i + \Delta p_{imk}, m) \Big]. \tag{4}$$

As illustrated in Fig. 2, iDETR replaces the standard deformable attention in DETR-based models with iAttn.

**Implementation of the local INR function in Eq. (3).** To facilitate optimization, the local INR function $\text{INRLCS}(\cdot)$ is parameterized as a lightweight multi-layer perceptron (MLP). The hypernetwork $\Phi(\cdot)$ is also an MLP that generates the parameters $\theta$ for $\text{INRLCS}(\cdot)$. Drawing inspiration from low-rank adaptation techniques [37, 38], we constrain $\Phi(\cdot)$ to produce only two low-rank factor matrices $A \in \mathbb{R}^{d_{\text{out}} \times r}$ and $B \in \mathbb{R}^{r \times d_{\text{in}}}$ for the $l$-th layer. These are then combined to form the low-rank update to the weight matrix:

$$W^l = A^l \times B^l + W^l_s, \tag{5}$$

where $W^l_s$ denotes the shared base weight matrix common to all local INR functions. This low-rank decomposition significantly reduces the number of learnable parameters in the hypernetwork while maintaining expressive power.

To further improve efficiency, all iAttn modules across the decoder layers share a single instance of the unified implicit sampler $f_{\text{Inrcs}}(\cdot)$. For any given input image, the sampler weights are computed only once and reused throughout the decoder. Additionally, to standardize the input coordinates to INRLCS$(\cdot)$, we normalize the 2D query positions such that the image domain spans $[0, 2W]$ in width and $[0, 2H]$ in height. This ensures that the relative offsets $p - p^*$ remain bounded within $[-1, 1]$, improving training stability.

**More residual connections.** Although iAttn enables feature querying at arbitrary continuous positions, some spatial information is inevitably lost during backbone forward propagation and downsampling, which can limit the fidelity of the learned local INR functions. To mitigate this and provide richer spatial cues for constructing accurate implicit representations, we introduce residual connections from earlier backbone stages. Formally, let $\{\mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4, \mathbf{S}_5\}$ denote the feature maps from the 4 different stages of the backbone, with downsampling rates of $\{4, 8, 16, 32\}$, respectively. It is worth noting that iDETR only leverages $\mathbf{S}_5$ for INR construction. Residual connections are introduced from $\mathbf{S}_2$, $\mathbf{S}_3$, and $\mathbf{S}_4$ directly to $\mathbf{S}_5$, where $\mathbf{S}_2$, $\mathbf{S}_3$, and $\mathbf{S}_4$ are spatially $8\times$, $4\times$, and $2\times$ larger than $\mathbf{S}_5$, respectively. Here, features from each early stage are aligned to the spatial resolution and channel dimension of $\mathbf{S}_5$ using a $1 \times 1$ convolution followed by a pixel unshuffle (inverse pixel shuffle) operation. Compared to standard upsampling or interpolation methods, pixel unshuffle better preserves spatial details with virtually no additional computational overhead.

### 3.2.2 CGQI: Centroid-Guided Query Initialization

In conventional DETR-based models, object queries are typically initialized from the top-$k$ features selected across multi-scale encoder outputs. However, since iDETR operates solely on a single-scale low-resolution feature map $\mathbf{F}_5$, two critical challenges arise during query initialization: (1) Multi-scale designs naturally leverage scale-specific anchor boxes to cover objects of varying sizes; with only one scale available, manually designing effective anchors becomes impractical and suboptimal. (2) The lack of higher-resolution features (e.g., $F_2$, $F_3$) severely limits the number and diversity of candidate features for selecting high-quality initial queries.

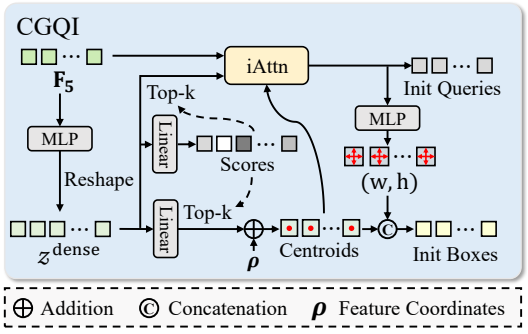



Figure 3: Pipeline of CGQI. CGQI first generates candidate proposals, then predicts the centroids of the top-$k$ proposals, and applies iAttn to refine features around these centroids, yielding high-quality initial queries and bounding boxes.

To overcome these limitations, we propose **Centroid-Guided Query Initialization (CGQI)**. As illustrated in Fig. 3, CGQI first expands the feature dimension of $\mathbf{F}_5$ from $d$ to $d \cdot c$ using a lightweight MLP, then reshapes the output into a dense feature map $z^{\text{dense}} \in \mathbb{R}^{HW \cdot c \times d}$. This dense expansion allows each latent code $z_{i,j}$ in $\mathbf{F}_5$ to be associated with multiple candidate representations, effectively increasing proposal density.

Following standard practice, a linear layer is applied to predict confidence scores for these dense features, from which the top-$k$ proposal features $z_i^p$ are selected. Inspired by anchor-free detection paradigms, we abandon predefined anchors and decouple query initialization into two steps: center-point prediction and box-size regression.

Center points are predicted directly from the selected proposal features via a linear layer:

$$p_i^c = \text{Linear}(z_i^p) + \rho_i, \tag{6}$$

where $\rho_i$ denotes the grid coordinate of the corresponding latent code $z_i^p$.

Unlike prior anchor-free approaches, CGQI introduces an additional **iAttn** module to refine the proposal feature $z_i^p$ around the predicted center $p_i^c$ before box-size regression. This "look-twice" mechanism enables the model to re-query and aggregate finer-grained continuous features from the implicit field, yielding more semantically rich initial query embeddings $q_i^{(0)}$:

$$q_i^{(0)} = f_{\text{IMA}}(z_i^p, p_i^c, \mathbf{F}_5). \tag{7}$$

Finally, the width and height of each initial query are regressed directly from the refined query feature $q_i^{(0)}$ using an MLP:

$$p_i^{wh} = \text{MLP}(q_i^{(0)}). \tag{8}$$

# 4 Experiments

Table 1: Experimental results on AI-TODv2. All models are trained on the train+val split and evaluated on the test set. The † denotes multi-scale variants using $\mathbf{S}_2$-$\mathbf{S}_5$ for enhanced TOD performance. The best result is in **bold**, and the second best is underlined.

| Model | #Para | GFlops | AP | $AP_{50}$ | $AP_{75}$ | $AP_{vt}$ | $AP_t$ | $AP_s$ | $AP_m$ |
|---|---|---|---|---|---|---|---|---|---|
| **CNN-Based Models** | | | | | | | | | |
| CascadeR-CNN [39] | 68.9M | 88.7 | 15.1 | 34.2 | 11.2 | 0.1 | 11.5 | 26.7 | 38.5 |
| DetectoRS [3] | 134.8M | $\approx 160$ | 16.1 | 35.5 | 12.5 | 0.1 | 12.6 | 28.3 | 40.0 |
| DetectoRS w/ NWD [40] | - | - | 20.8 | 49.3 | 14.3 | 6.4 | 19.7 | 29.6 | 38.3 |
| DetectoRS w/ RFLA [24] | - | - | 24.8 | 55.2 | 18.5 | 9.3 | 24.8 | 30.3 | 38.2 |
| DNTR [41] | 128.4M | 298.1 | 26.2 | 56.7 | 20.2 | 12.8 | 26.4 | 31.0 | 37.0 |
| **DETR-Based Models** | | | | | | | | | |
| Deformable-DETR [14] | 40.0M | 232.5 | 18.2 | 59.7 | 10.2 | 5.8 | 16.8 | 25.2 | 34.0 |
| DINO-DETR [4] | 47.0M | 338.6 | 25.6 | 61.3 | 17.4 | 12.7 | 25.3 | 31.7 | 39.4 |
| DQ-DETR [42] | 58.7M | 1805.4 | 30.2 | <u>68.6</u> | 22.3 | 15.3 | 30.5 | 36.5 | 44.6 |
| D-FINE-S† [5] | 11.6M | 112.4 | 30.1 | 64.1 | 25.5 | 14.0 | 30.4 | 36.0 | 45.4 |
| D-FINE-M† [5] | 22.3M | 211.4 | 30.8 | 65.4 | 25.3 | 14.3 | 30.1 | 37.6 | 46.8 |
| D-FINE-L† [5] | 34.0M | 327.5 | 31.3 | 66.7 | 24.9 | 15.3 | 31.0 | 38.4 | 46.6 |
| Dome-DETR-S [21] | 13.2M | 154.2 | 33.3 | 67.5 | 28.9 | 17.8 | 33.0 | 38.4 | 46.4 |
| Dome-DETR-M [21] | 23.9M | 252.6 | 34.0 | 68.4 | 29.9 | 18.4 | 34.3 | 39.0 | 46.9 |
| Dome-DETR-L [21] | 36.0M | 358.7 | <u>34.6</u> | **69.2** | **32.0** | <u>19.0</u> | **35.6** | **39.8** | **47.3** |
| **iDETR (Ours)** | 36.6M | 106.0 | 30.7 | 64.4 | 24.6 | 16.7 | 31.0 | 35.6 | 44.0 |
| **iDETR-DC5 (Ours)** | 36.9M | 224.6 | **34.8** | 68.5 | <u>30.1</u> | **20.1** | <u>35.2</u> | <u>38.6</u> | <u>47.1</u> |

## 4.1 Experiment Setup

**Datasets.** Following [21], we conduct experiments on the AI-TODv2 [2] and VisDrone [43] datasets to verify the effectiveness of our proposed iDETR. Detailed descriptions of the datasets are provided in Appendix A.2.

**Implementation details.** Our method is built upon D-FINE-L [5]. We provide the step-by-step modifications of our model based on D-FINE-L in Appendix A.4. All models are trained on 4 RTX 5880 GPUs with a total batch size of 8. We use the AdamW optimizer for 120 training epochs. We set the base learning rate to 2e-4 and the backbone learning rate to 2e-5, with the base learning rate decayed by a factor of 0.8 at the 60th and 90th epochs. Following [21], we set the input resolution of both AI-TODv2 and VisDrone to $800 \times 800$ during training and testing.

We introduce two variants: **iDETR** and **iDETR-DC5**. The DC5 variant employs the dilated C5 backbone [13], a configuration commonly used in single-scale detectors. Hyperparameter details are provided in Appendix (see Sec. A.7).

**Evaluation metrics.** Following the standard COCO-style protocol, we evaluate model performance using Average Precision (AP). In addition to the conventional metrics, AP, $AP_{50}$, and $AP_{75}$, we also report size-specific APs as defined in the AI-TOD benchmark: $AP_{vt}$ (very tiny objects, 2–8 pixels), $AP_t$ (tiny objects, 8–16 pixels), $AP_s$ (small objects, 16–32 pixels), and $AP_m$ (medium objects, larger than 32 pixels). These fine-grained metrics allow a more comprehensive assessment of performance on extremely small objects, which is the primary focus of TOD.

## 4.2 Main Results

**Experiments on AI-TODv2.** Table 1 compares our proposed iDETR against the state-of-the-art CNN-based and DETR-based models on the AI-TODv2 dataset. The results demonstrate that

Table 2: Experimental results on VisDrone. All models are trained on the training set and evaluated on the validation set. The best result is in **bold**, and the second best is underlined.

| Model | #Para | GFlops | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|
| **CNN-Based Models** | | | | | |
| CascadeR-CNN [39] | 68.9M | 236.4 | 22.6 | 38.8 | 23.2 |
| DetectoRS w/ RFLA [24] | 134.8M | 160.0 | 27.4 | 45.3 | 23.9 |
| CEASC [44] | $\approx$ 90.0M | 150.2 | 28.7 | 50.7 | 24.7 |
| DNTR [41] | 128.4M | 373.4 | 33.1 | 53.8 | 34.8 |
| **DETR-Based Models** | | | | | |
| DQ-DETR [42] | 58.7M | 1782.4 | 35.2 | 54.9 | 35.6 |
| UAV-DETR [45] | 45.5M | 247.1 | 32.4 | 51.9 | 33.6 |
| D-FINE-M [5] | 22.3M | 211.4 | 33.6 | 56.9 | 36.7 |
| D-FINE-L [5] | 34.4M | 327.5 | 36.5 | 58.0 | 38.2 |
| Dome-DETR-M [21] | 23.9M | 284.6 | 36.1 | 59.8 | 39.4 |
| Dome-DETR-L [21] | 36.0M | 376.4 | <u>39.0</u> | <u>61.1</u> | <u>40.8</u> |
| **iDETR-DC5 (Ours)** | 37.2M | 219.5 | 36.9 | 59.4 | 38.1 |
| **Dome-DETR-M w/ iAttn (Ours)** | 24.8M | 313.3 | 38.3 | 60.7 | 39.9 |
| **Dome-DETR-L w/ iAttn (Ours)** | 36.9M | 407.9 | **39.4** | **62.0** | **41.9** |

iDETR achieves superior performance and significantly higher efficiency. Specifically, iDETR-DC5 outperforms the second-best method by 0.2% AP, while reducing the computational cost by 134.1 GFLOPs (37%) during inference. Furthermore, our model achieves a more substantial gain on very tiny objects, surpassing the runner-up by 1.1% AP.

**Experiments on VisDrone.** Table 2 compares our proposed iDETR against the state-of-the-art models on the VisDrone dataset. Since objects in the VisDrone dataset are generally relatively large, iDETR cannot fully utilize its advantage in detecting tiny objects. Consequently, the overall performance of iDETR-DC5 slightly lags behind the state-of-the-art method. Nevertheless, compared to the efficiency-comparable model Dome-DETR-M and the baseline D-FINE-L, our method achieves better performance (36.9% AP vs. 36.1% and 36.5%). Furthermore, integrating the proposed iAttn module into Dome-DETR-L yields state-of-the-art detection performance (+0.4% AP compared to Dome-DETR-L), which demonstrates the generalizability of our proposed iAttn module.

**iAttn as a plug-and-play module.** Our proposed iAttn can serve as a substitute for deformable attention and theoretically replace the deformable attention layer in the decoder of any DETR-based model. In Table 3, we compare the performance of three different DETR variants equipped with iAttn. The results demonstrate that embedding iAttn yields consistent AP gains across all models, with more prominent improvements on very tiny objects. Specifically, integrating iAttn brings consistent performance gains of +2.2%, +1.1%, and +0.2% in AP to the baseline models. For very tiny objects, the corresponding performance improvements attain +4.6%, +2.7%, and +3.1% in $AP_{vt}$, respectively.

Table 3: Performance of DETR-based models with and without iAttn on AI-TODv2.

| Model | AP | $AP_{vt}$ | $AP_t$ | $AP_s$ | $AP_m$ |
|---|---|---|---|---|---|
| Deformable-DETR [14] | 18.2 | 5.8 | 16.8 | 25.2 | **34.0** |
| **+ iAttn (Ours)** | **20.4** | **10.4** | **21.0** | **25.3** | 29.3 |
| DINO-DETR [4] | 25.6 | 12.7 | 25.3 | 31.7 | **39.4** |
| **+ iAttn (Ours)** | **26.7** | **15.4** | **27.1** | **31.8** | 38.1 |
| Dome-DETR-L [21] | 34.6 | 19.0 | **35.6** | **39.8** | **47.3** |
| **+ iAttn (Ours)** | **34.8** | **22.1** | 35.2 | 38.9 | 46.0 |

**Discussion.** While iDETR achieves a better accuracy-efficiency trade-off and superior performance on tiny objects, it underperforms Dome-DETR for relatively large object detection. Moreover, iDETR only dominates at high IoU thresholds and is slightly surpassed by Dome-DETR under low IoU settings (see Appendix A.6). This suggests that iDETR has strong bounding box regression ability but may suffer from marginally lower recall performance. We attribute these drawbacks to two key factors: (1) **The sampling paradigm in iAttn remains discrete.** iAttn samples continuous features by generating multiple discrete sampling points. However, limited sampling points cannot effectively cover diverse spatial locations of large objects. (2) **Continuous feature learning lacks direct supervision.** Contin-

Table 4: Ablation study for our proposed iDETR on AI-TODv2.

| iAttn | CGQI | #Para | GFlops | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|---|
| ✘ | ✘ | 22.8M | 76.5 | 23.8 | 55.0 | 17.3 |
| ✓ | ✘ | 35.2M | 89.0 | 28.1 | 59.2 | 22.5 |
| ✘ | ✓ | 24.1M | 88.6 | 25.4 | 57.6 | 19.0 |
| ✓ | ✓ | 36.6M | 106.0 | **30.7** | **64.4** | **24.6** |

Table 5: LIIF vs. our INR-based method. Only INR computational costs are reported.

| Method | GFlops | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|
| LIIF [46] | 29.5 | 29.6 | 62.8 | 23.5 |
| $LIIF_{le}$ [46] | 82.2 | 30.0 | 64.1 | 24.0 |
| iDETR | **7.2** | **30.7** | **64.4** | **24.6** |
| $iDETR_{le}$ | 26.1 | 30.0 | 63.5 | 23.7 |

uous feature learning is only implicitly optimized via iAttn queries, leading to weak representations for some object regions.

### 4.3 Ablation Study

We conduct extensive ablations on the AI-TODv2 dataset to validate the effectiveness of iAttn and CGQI. Note that ablation experiments here are conducted with iDETR by default. More ablations are provided in Appendix (see Sec. A.5).

**Effectiveness of individual components.** Table 4 shows the contribution of each module. Replacing deformable attention with iAttn yields a +4.3% AP gain. Adding CGQI to the baseline improves AP by 1.6%. Combining both components (full iDETR) results in a total gain of +6.9% AP. The results confirm that our proposed iAttn and CGQI effectively enhance tiny object detection performance.

**Comparison of INR-based sampling methods.** We compare our INR-based sampling in iAttn against LIIF [46], adapted for feature retrieval with head-conditional sampling for fairness. Table 5 reports results, including $LIIF_{le}$ (with local ensemble [46]). Our iDETR outperforms LIIF across all metrics while incurring only 1/4 of the computational cost. Adding a local ensemble to our method yields no further gain, indicating it is unnecessary for iAttn.

**Effectiveness of head-conditional sampling.** Head-conditional sampling integrates head-specific projections directly into the INR sampler. Table 6 compares variants with and without it. The proposed design reduces computational cost by 4×, memory footprint by 3×, and simultaneously improves AP by 0.8%, demonstrating strong efficiency and performance benefits.

Table 6: Ablation on head-conditional sampling (AI-TODv2). Here, we report the computational cost of INR-based functions only. Memory footprints are measured at $800 \times 800$ input resolution and batch size 1. "HCS" denotes the proposed head-conditional sampling.

| Method | #Para | GFlops | Memory | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|---|
| w/o HCS | 36.9M | 28.2 | 9.7 GB | 29.9 | 62.9 | 23.7 |
| w/ HCS | 36.6M | **7.2** | **3.1 GB** | **30.7** | **64.4** | **24.6** |

### 4.4 Visualization Analysis

To further investigate whether iDETR learns continuous implicit features, the learned features are visualized at pixel-level resolution. Specifically, the INR-based sampler is queried at each pixel coordinate, and the resulting head-specific features are concatenated. We then apply principal component analysis (PCA) to visualize these features in RGB. As shown in Figure 4, the implicit features reveal recognizable object structures and spatial extents across diverse scenes. In particular, they can distinguish closely spaced tiny objects in dense scenes, highlighting their ability to preserve fine-grained spatial information. However, the implicit features still exhibit noticeable noise in some high-frequency background regions, suggesting difficulties in modeling complex local patterns.

## 5 Conclusion

In this work, we propose iDETR, an efficient detection transformer that addresses the discrete representation and computational bottlenecks of conventional multi-scale feature pyramids for tiny

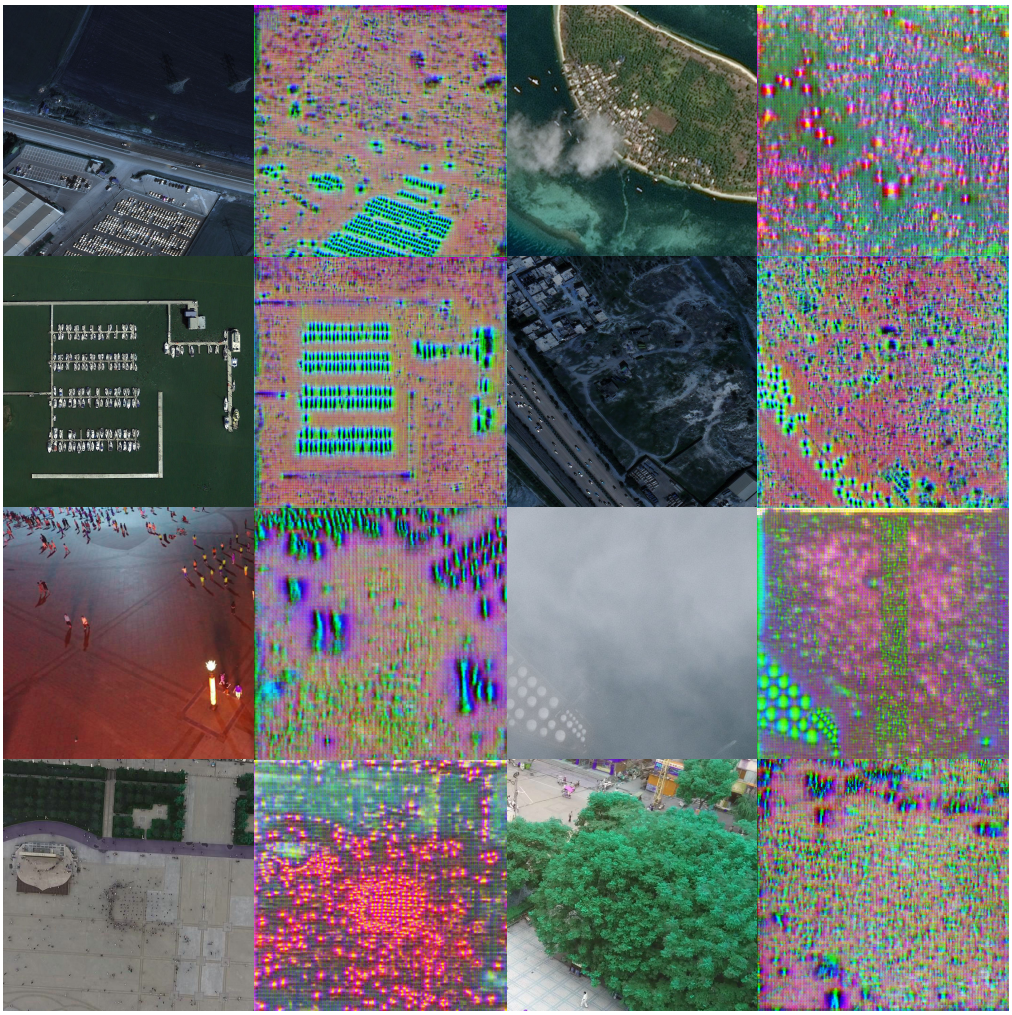



Figure 4: Visualization of implicit features. Columns (a) and (c) show the original images in dense and sparse scenes, respectively, while columns (b) and (d) display their corresponding implicit features visualized using PCA.

object detection. By abandoning discrete multi-scale features and learning continuous implicit representations solely from low-resolution single-scale inputs, iDETR achieves competitive TOD performance with substantially reduced computational overhead. **Limitations and future work:** However, iDETR still suffers from discrete sampling and insufficient direct supervision for continuous features. In future work, we aim to address these limitations and explore extending the single-scale continuous feature paradigm to other multi-scale models.

## 6 Acknowledgment

This work was partially supported by the National Key Research and Development Program of China (2024YFF0619800), the National Natural Science Foundation of China (No.62672333 and No.62473267), the Special Subject on Agriculture and Social Development, Key Research and Development Plan in Guangzhou (No.2025B03J0019), the Guangdong Basic and Applied Basic Research Foundation (No.2025A1515011614), the Natural Science Foundation of Top Talent of SZTU (No.GDRC202318), and the Innovative Team Project in Guangdong Province (No. 2025KCXTD040). The authors declare no competing interests.

## References


[1] J. Wang, W. Yang, H. Guo, R. Zhang, and G.-S. Xia, "Tiny object detection in aerial images," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 3791–3798.

[2] C. Xu, J. Wang, W. Yang, H. Yu, L. Yu, and G.-S. Xia, "Detecting tiny objects in aerial images: A normalized wasserstein distance and a new benchmark," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 190, pp. 79–93, 2022.

[3] S. Qiao, L.-C. Chen, and A. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 213–10 224.

[4] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. Ni, and H.-Y. Shum, "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," in *International Conference on Learning Representations*, 2023.

[5] Y. Peng, H. Li, P. Wu, Y. Zhang, X. Sun, and F. Wu, "D-fine: Redefine regression task of detrs as fine-grained distribution refinement," in *International Conference on Learning Representations*, 2025.

[6] 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*, 2025, pp. 15 162–15 171.

[7] Y. Tian, Q. Ye, and D. Doermann, "Yolov12: Attention-centric real-time object detectors," in *Advances in Neural Information Processing Systems*, 2025.

[8] M. Lei, S. Li, Y. Wu, H. Hu, Y. Zhou, X. Zheng, G. Ding, S. Du, Z. Wu, and Y. Gao, "Yolov13: Real-time object detection with hypergraph-enhanced adaptive visual perception," *arXiv preprint arXiv:2506.17733*, 2025.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.

[10] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.

[11] H. Li, "Rethinking features-fused-pyramid-neck for object detection," in *European Conference on Computer Vision*. Springer, 2024, pp. 74–90.

[12] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9627–9636.

[13] 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*. Springer, 2020, pp. 213–229.

[14] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," in *International Conference on Learning Representations*, 2021.

[15] F. Li, H. Zhang, S. Liu, J. Guo, L. M. Ni, and L. Zhang, "Dn-detr: Accelerate detr training by introducing query denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 619–13 627.

[16] Z. Zong, G. Song, and Y. Liu, "Detrs with collaborative hybrid assignments training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6748–6758.

[17] Q. Chen, X. Chen, J. Wang, S. Zhang, K. Yao, H. Feng, J. Han, E. Ding, G. Zeng, and J. Wang, "Group detr: Fast detr training with group-wise one-to-many assignment," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 6633–6642.

[18] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen, "Detrs beat yolos on real-time object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 965–16 974.

[19] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[20] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 781–10 790.

[21] Z. Hu, P. Wu, J. Chen, H. Zhu, Y. Wang, Y. Peng, H. Li, and X. Sun, "Dome-detr: Detr with density-oriented feature-query manipulation for efficient tiny object detection," in *Proceedings of the 33rd ACM International Conference on Multimedia*, 2025, pp. 101–110.

[22] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, and K. Cho, "Augmentation for small object detection," in *9th International Conference on Advances in Computing and Information Technology (ACITY 2019)*. Aircc Publishing Corporation, 2019.

[23] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9759–9768.

[24] C. Xu, J. Wang, W. Yang, H. Yu, L. Yu, and G.-S. Xia, "Rfla: Gaussian receptive field based label assignment for tiny object detection," in *European Conference on Computer Vision*. Springer, 2022, pp. 526–543.

[25] B. Cao, H. Yao, P. Zhu, and Q. Hu, "Visible and clear: Finding tiny objects in difference map," in *European Conference on Computer Vision*. Springer, 2024, pp. 1–18.

[26] H. Sun, R. Wang, Y. Li, L. Yang, S. Lin, X. Cao, and B. Zhang, "Set: Spectral enhancement for tiny object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025, pp. 4713–4723.

[27] J. Bian, M. Feng, W. Dong, F. Wu, J. Luo, Y. Wang, and G. Shi, "Feature information driven position gaussian distribution estimation for tiny object detection," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 30 376–30 386.

[28] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.

[29] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser *et al.*, "Local implicit grid representations for 3d scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6001–6010.

[30] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision*. Springer, 2020, pp. 405–421.

[31] Z. Wang and D. Xu, "Hyrf: Hybrid radiance fields for memory-efficient and high-quality novel view synthesis," in *Advances in Neural Information Processing Systems*, 2025.

[32] S. Gao, X. Liu, B. Zeng, S. Xu, Y. Li, X. Luo, J. Liu, X. Zhen, and B. Zhang, "Implicit diffusion models for continuous super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 10 021–10 030.

[33] S. R. Jyhne, C. Igel, M. Goodwin, P.-A. Andersen, S. Belongie, and N. Lang, "Superf: Neural implicit fields for multi-image super-resolution," *arXiv preprint arXiv:2512.09115*, 2025.

[34] L. Chai, M. Gharbi, E. Shechtman, P. Isola, and R. Zhang, "Any-resolution training for high-resolution image synthesis," in *European Conference on Computer Vision*. Springer, 2022, pp. 170–188.

[35] M. Gwilliam, X. Wang, X. Hu, and Z. Yang, "Implicit neural representation facilitates unified universal vision encoding," *arXiv preprint arXiv:2601.14256*, 2026.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[37] I. Skorokhodov, S. Ignatyev, and M. Elhoseiny, "Adversarial generation of continuous images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 753–10 764.

[38] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *International Conference on Learning Representations*, vol. 1, no. 2, p. 3, 2022.

[39] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6154–6162.

[40] J. Wang, C. Xu, W. Yang, and L. Yu, "A normalized gaussian wasserstein distance for tiny object detection," *arXiv preprint arXiv:2110.13389*, 2021.

[41] H.-I. Liu, Y.-W. Tseng, K.-C. Chang, P.-J. Wang, H.-H. Shuai, and W.-H. Cheng, "A denoising fpn with transformer r-cnn for tiny object detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–15, 2024.

[42] Y.-X. Huang, H.-I. Liu, H.-H. Shuai, and W.-H. Cheng, "Dq-detr: Detr with dynamic query for tiny object detection," in *European Conference on Computer Vision*. Springer, 2024, pp. 290–305.

[43] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling, "Detection and tracking meet drones challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7380–7399, 2021.

[44] B. Du, Y. Huang, J. Chen, and D. Huang, "Adaptive sparse convolutional networks with global context enhancement for faster object detection on drone images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 435–13 444.

[45] H. Zhang, H. Zhang, K. Liu, Z. Gan, and G.-N. Zhu, "Uav-detr: efficient end-to-end object detection for unmanned aerial vehicle imagery," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 15 143–15 149.

[46] Y. Chen, S. Liu, and X. Wang, "Learning continuous image representation with local implicit image function," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8628–8638.

[47] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[48] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020.

[49] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7462–7473, 2020.

[50] I. Skorokhodov, S. Ignatyev, and M. Elhoseiny, "Adversarial generation of continuous images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 753–10 764.

# A Appendix

## A.1 Usage of LLMs

We use large language models (LLMs) solely for polishing the writing, such as improving sentence fluency and fixing grammar issues. No content generation, idea creation, or experimental analysis is conducted by LLMs.

## A.2 Datasets

**AI-TODv2.** This dataset comprises 28,036 aerial images, divided into 11,214 training, 2,804 validation, and 14,018 test images. It includes a total of 752,745 annotated object instances across 8 categories. Objects in this dataset are extremely small, with an average size of only 12.7 pixels and 86% of instances smaller than 16 pixels, making it one of the most challenging benchmarks for tiny object detection.

**VisDrone.** This is a widely used drone-captured detection dataset, containing 6,471 training, 548 validation, and 3,190 test images. It covers 10 common object categories and features diverse scenes with sparse and crowded distributions. Following existing protocols, we train on the training set and evaluate on the validation set.

## A.3 Experimental Computing Resources

All experiments are conducted using Python 3.11 and PyTorch 2.6.0. The training and validation are performed on a workstation with 8× NVIDIA RTX 5880 Ada Generation GPUs (48GB VRAM), an Intel Xeon Gold 6530 CPU, and 1024GB system memory. Table 7 reports the training time and peak GPU memory consumption.

Table 7: Compute resources of iDETR during training. Here, Memory refers to the peak GPU memory consumption per single GPU during training, with a per-GPU batch size of 2 and a total batch size of 8 across 4 GPUs.

| Model | Dataset | Memory | Time |
|---|---|---|---|
| iDETR | AI-TODv2 | 28.1 GB | 17.0 h |
| iDETR-DC5 | AI-TODv2 | 30.1 GB | 29.4 h |
| iDETR-DC5 | VisDrone | 30.2 GB | 15.5 h |

## A.4 Step-by-step Modifications from D-FINE

In Table 8, we present the detailed stepwise modifications from D-FINE-L to iDETR-DC5. We also report the number of parameters, computational complexity, and AP performance at each step. Starting from the baseline D-FINE-L, removing the multi-scale design drastically reduces GFlops from 327.5 to 76.5 but causes a noticeable AP drop from 31.3% to 23.8%. Equipping the model with the proposed iAttn and CGQI modules raises the computational complexity to 106.0 GFlops and recovers the AP to 30.7%. Further introducing the dilated C5 backbone increases GFlops to 174.1 and boosts the AP to 33.0%.

In iDETR, each latent code in the feature map is constructed into an individual INR function. However, a single INR function has limited representation capacity and tends to converge poorly under limited supervision. To address this issue, we propose **Denser INR Functions**, which construct a denser family of INR functions by upsampling the feature map. Specifically, we adopt PixelShuffle to upsample the feature map and then utilize the upsampled features to generate a family of local INR-based functions. Incorporating Denser INR Functions further pushes GFlops to 224.4. Nevertheless, directly feeding the upsampled features into the global-attention-based encoder causes a sharp surge in memory usage. To address this issue, we further introduce window attention [47]. Replacing the AIFI [18] in the encoder with window attention maintains a comparable computational overhead while obtaining the best AP of 34.8%.

Table 8: Step-by-step modifications from baseline model D-FINE-L to iDETR-DC5 on AI-TODv2. "O/M" indicates the "out-of-memory" error.

| Model | #Para | GFlops | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|
| Baseline: D-FINE-L [5] | 34.0M | 327.5 | 31.3 | 66.7 | 24.9 |
| Remove Multi-scale Design | 22.8M | 76.5 | 23.8 | 55.0 | 17.3 |
| + iAttn & CGQI | 36.6M | 106.0 | 30.7 | 64.4 | 24.6 |
| + dilated C5 backbone | 36.8M | 174.1 | 33.0 | 66.4 | 27.9 |
| + Denser INR Functions | 36.8M | 224.4 | O/M | O/M | O/M |
| Encoder AIFI $\rightarrow$ Window Attention [47] | 36.9M | 224.6 | 34.8 | 68.5 | 30.1 |

### A.5 More Ablations

**Impact of rank $r$ in iAttn.** We examine the low-rank parameter generation with rank $r$. As shown in Table 9, performance improves with larger $r$ and saturates at $r = 4$. Increasing $r$ incurs negligible additional cost, since $A$ and $B$ are generated only once per inference.

Table 9: Performance comparison of models with different rank $r$ on AI-TODv2.

| Rank | #Para | GFlops | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|
| $r = 1$ | 36.5M | 102.1 | 29.5 | 61.7 | 24.0 |
| $r = 4$ | 36.6M | 106.0 | **30.7** | **64.4** | 24.6 |
| $r = 8$ | 36.7M | 111.2 | 30.3 | 63.9 | **24.8** |
| $r = 12$ | 36.9M | 116.4 | 30.1 | 63.3 | 24.0 |

**Number of candidate features in CGQI.** We study the number $c$ of candidate features generated per latent code. Table 10 shows optimal performance at $c = 16$.

Table 10: Performance comparison of models with different number of candidate features $c$ on AI-TODv2.

| Number | #Para | GFlops | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|
| $c = 4$ | 36.4M | 105.8 | 30.1 | 63.6 | 24.3 |
| $c = 9$ | 36.5M | 105.9 | 30.2 | 63.2 | 24.3 |
| $c = 16$ | 36.6M | 106.0 | **30.7** | **64.4** | 24.6 |
| $c = 25$ | 36.7M | 106.2 | 30.6 | 64.1 | **24.8** |
| $c = 36$ | 36.9M | 106.5 | 30.1 | 63.5 | 24.1 |

**Effectiveness of residual connections.** To better construct the INR function, we incorporate more residual connections into the backbone. Table 11 compares the variants with and without it. After adding the residual connections, iDETR achieves an improvement of 2.5% AP.

**Effectiveness of position encodings.** Some studies have demonstrated that positional encoding can effectively enhance the representation capacity of INR [48, 49, 50]. Following [50], we apply a linear matrix $W \in \mathbb{R}^{n\times 2}$ to the coordinate $p - p^*$, followed by nonlinearities of sine/cosine. The INR function with positional encoding can be formalized as:

$$f_I(F, p, m) = f_l(\text{concat}(\sin(W(p - p^*)), \cos(W(p - p^*))), \gamma(m); \theta_{i^*,j^*}). \tag{9}$$

As shown in Table 12, iDETR achieves an improvement of 0.4% AP after introducing positional encodings.

**Effectiveness of the architecture of INR functions.** In iDETR, we implement the INR function as a lightweight multi-layer perceptron (MLP) network. Here, we further investigate the impact of different MLP architectures on the INR function. The lightweight MLP used in iDETR by default has one hidden layer with 32 neurons. To investigate the impact of different MLP architectures on the INR function, we design three other distinct MLP structures: (1) MLP with deeper layers, (2)

Table 11: Performance comparison of models with and without residual connections. "MRC" denotes the use of additional residual connections.

| Setting | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| w/o MRC | 28.2 | 61.8 | 21.3 |
| w/ MRC | **30.7** | **64.4** | **24.6** |
| Improvement | +2.5 | +2.6 | +3.3 |

Table 12: Performance comparison of models with and without position encodings. "PE" denotes position encodings.

| Setting | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| w/o PE | 30.3 | 63.9 | **24.8** |
| w/ PE | **30.7** | **64.4** | 24.6 |
| Improvement | +0.4 | +0.5 | -0.2 |

MLP with more neurons in the hidden layer, and (3) a single-layer MLP (which is essentially a linear layer).

As shown in Table 13, compared to the use of heavier MLP architectures, iDETR equipped with only a lightweight MLP with one hidden layer (hidden dimension 32) achieves better performance (30.7% AP). When employing a linear layer to model the INR function, its performance is inferior to that of the lightweight MLP due to its limited fitting capability.

Table 13: Effectiveness of different MLP architectures on INR function performance in iDETR. The number inside the square bracket denotes the dimension of each hidden layer.

| MLP Architecture | #Para | GFlops | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|
| [32] | 36.6M | 106.0 | **30.7** | **64.4** | 24.6 |
| [32, 32] | 36.7M | 109.1 | 30.4 | 63.4 | 24.7 |
| [64] | 36.7M | 112.4 | 30.6 | 63.6 | **24.9** |
| Linear | 36.5M | 103.0 | 30.4 | 64.0 | 24.4 |

**Effectiveness of the number of heads.** As shown in Table 14, the performance of iDETR improves significantly with an increase in the number of heads, $M$. However, increasing $M$ also requires larger training GPU memory.

**Effectiveness of the number of sampling points.** The number of sampling points determines how many implicit features at distinct locations each query can retrieve. In Tab. 15, we investigate the performance of iDETR-DC5 under different numbers of sampling points.

**Ablation on CGQI.** CGQI acquires high-quality initial queries through two key designs: proposal densification and the re-query mechanism. To further study their individual effectiveness, we conduct ablation experiments based on iDETR-DC5. As shown in Table 16, proposal densification alone improves AP by +1.1%, and the re-query mechanism alone brings a +0.8% AP gain. These results verify that both components can effectively enhance detection performance.

### A.6 More Visualization

**Comparison with the state-of-the-art TOD methods.** Figure 5 compares the detection results among D-FINE-L, Dome-DETR-L, and iDETR-DC5. The first two columns show detection predictions in dense scenes, while the last two columns show predictions in sparse scenes.

**CGQI visualization.** We further visualize the query initialization results from CGQI. As shown in Figure 6, nearly all objects are assigned one or more initial queries located close to their ground-truth locations, confirming that CGQI generates accurate, well-aligned initial queries for the decoder.

**AP visualization.** In Figure 7, we visualize the comparison of AP performance across IoU thresholds from 50% to 95% between iDETR-DC5 and Dome-DETR-L. In the range of $AP_{50}$–$AP_{80}$, iDETR-DC5 achieves overall comparable performance though slightly lower than Dome-DETR-L. By contrast, iDETR-DC5 substantially outperforms Dome-DETR-L at $AP_{85}$–$AP_{95}$.

Table 14: Effectiveness of the number of heads ($M$) on iDETR performance. "O/M" indicates the "out-of-memory" error.

| $M$ | #Para | GFlops | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|
| 1 | 36.7M | 103.0 | 29.6 | 62.4 | 23.2 |
| 4 | 36.5M | 104.1 | 30.4 | 63.6 | 24.5 |
| 8 | 36.6M | 106.0 | **30.7** | **64.4** | **24.6** |
| 16 | 36.8M | 109.8 | O/M | O/M | O/M |

Table 15: Performance comparison of models with different number of sampling points on AI-TODv2. "O/M" indicates the "out-of-memory" error.

| #Sampling Points | #Para | GFlops | AP | $AP_{vt}$ | $AP_t$ | $AP_s$ | $AP_m$ |
|---|---|---|---|---|---|---|---|
| 1 | 36.8M | 221.6 | 33.8 | 19.9 | 34.5 | 37.9 | 46.1 |
| 2 | 36.8M | 222.6 | 34.2 | 20.0 | 34.8 | 38.2 | 47.4 |
| 4 | 36.9M | 224.6 | **34.8** | **20.1** | 35.2 | **38.6** | 47.1 |
| 8 | 37.1M | 228.7 | 34.2 | 19.3 | 35.0 | 38.5 | 48.0 |
| 16 | 37.4M | 236.7 | 34.5 | 19.8 | **35.4** | **38.6** | **48.2** |
| 32 | 37.9M | 252.9 | O/M | O/M | O/M | O/M | O/M |

Table 16: Ablation study for our proposed CGQI on AI-TODv2.

| Proposal Densification | Re-query Mechanism | #Para | GFlops | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|---|
| ✘ | ✘ | 35.7M | 184.4 | 33.1 | 67.4 | 28.4 |
| ✘ | ✓ | 35.7M | 185.3 | 33.9 | 66.7 | 29.3 |
| ✓ | ✘ | 36.8M | 221.9 | 34.2 | 67.9 | 29.8 |
| ✓ | ✓ | 36.9M | 224.6 | **34.8** | **68.5** | **30.1** |

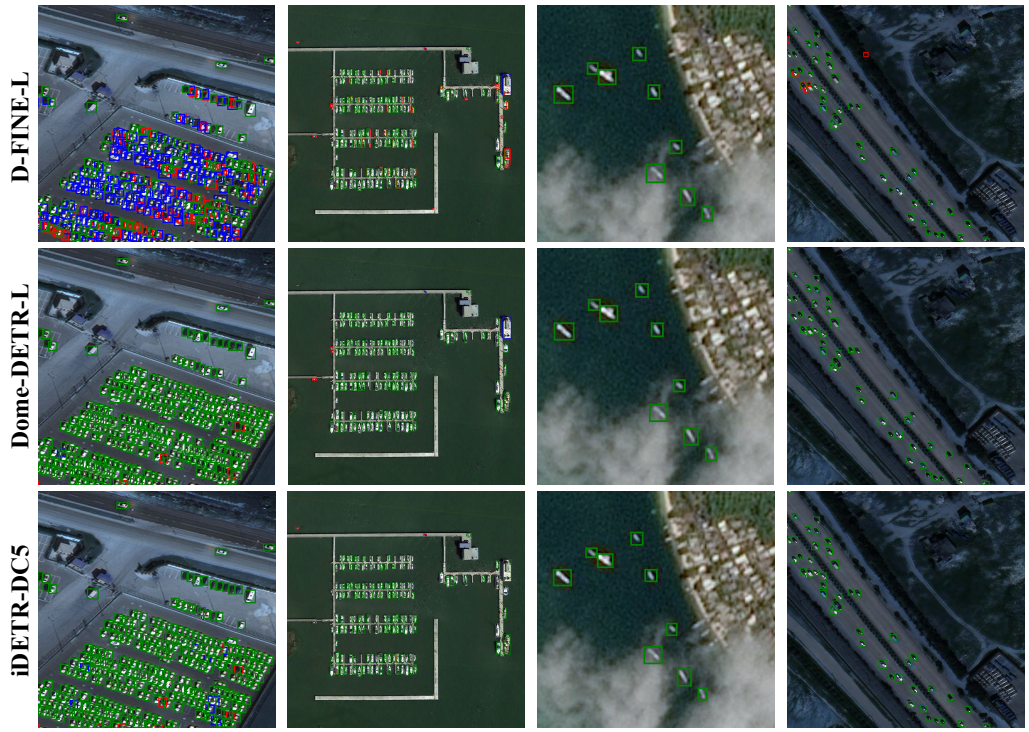



Figure 5: Detection performance comparison of D-FINE-L, Dome-DETR-L, and iDETR-DC5 in dense and sparse scenarios. Green boxes = true positives (TP), red boxes = false positives (FP), blue boxes = false negatives (FN).

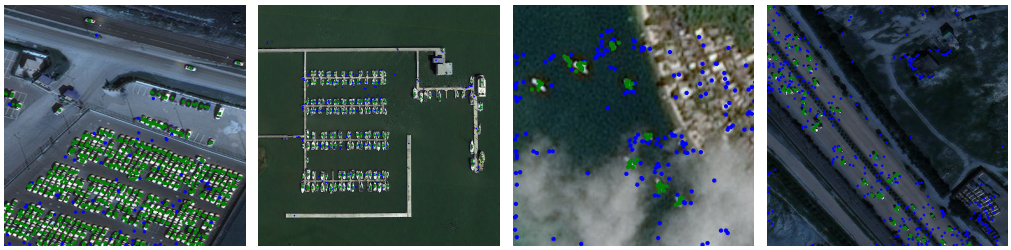


Figure 6: Query initialization results of CGQI on the AI-TODv2. Here, green dots denote queries correctly initialized within object bounding boxes, and blue dots represent those initialized outside.

In the range of $AP_{50}$–$AP_{80}$, the IoU threshold is relatively loose. Detection performance in this regime is primarily governed by object recall rather than fine localization accuracy, where coarsely predicted bounding boxes can still successfully match the ground truth. The slight underperformance of iDETR-DC5 in this interval implies that the model exhibits a relatively lower recall capability. By contrast, the stringent IoU criteria of $AP_{85}$–$AP_{95}$ place high demands on localization precision, boundary refinement and feature granularity. Even minor coordinate offsets can lead to obvious performance drops. This phenomenon further demonstrates that iDETR-DC5 enjoys superior capability in accurate bounding box regression.

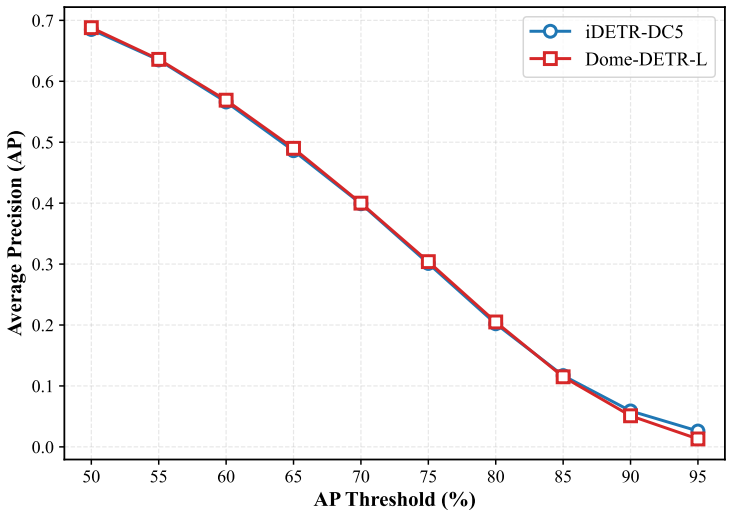



Figure 7: Comparison of AP performance across IoU thresholds from 50% to 95% between iDETR-DC5 and Dome-DETR-L.

### A.7 Hyperparameter Configurations for iDETR

Table 17 lists the detailed experimental hyperparameters of iDETR and iDETR-DC5 on the AI-TODv2 dataset. Both models adopt the same backbone HGNetv2-B4 and AdamW optimizer, with consistent configurations in embedding dimension, feedforward dimension, decoder layers, query number, sampling points, learning rates, weight decay, loss function weights, batch size, EMA decay and training epochs. The key differences lie in two customized settings: iDETR uses 16 candidate features and AIFI encoder attention without Denser INR Functions, whereas iDETR-DC5 adopts only 4 candidate features, replaces the encoder attention with window attention, and leverages Denser INR Functions. Other network dimensions for implicit modules are kept identical for both variants.

Table 18 presents the experimental configurations on the VisDrone dataset. Most hyperparameters remain consistent with those in Table 17, and the internal configuration differences between iDETR and iDETR-DC5 follow the same design as on AI-TODv2. The only dataset-specific adjustments are the number of object queries and sampling points, which are set to 500 and 12 respectively for both models on VisDrone.

Table 17: Experiment settings of iDETR and iDETR-DC5 on AI-TODv2.

| Setting | iDETR | iDETR-DC5 |
|---|---|---|
| Backbone Name | HGNetv2-B4 | HGNetv2-B4 |
| Optimizer | AdamW | AdamW |
| Embedding Dimension | 256 | 256 |
| Feedforward Dimension | 1024 | 1024 |
| Decoder Layers | 6 | 6 |
| Queries | 900 | 900 |
| Sampling Point Number | 4 | 4 |
| Base LR | 2e-4 | 2e-4 |
| Backbone LR | 2e-5 | 2e-5 |
| Weight Decay | 6.5e-5 | 6.5e-5 |
| Weight of $\mathcal{L}_{VFL}$ | 1 | 1 |
| Weight of $\mathcal{L}_{BBox}$ | 5 | 5 |
| Weight of $\mathcal{L}_{GIOU}$ | 2 | 2 |
| Weight of $\mathcal{L}_{FGL}$ | 0.15 | 0.15 |
| Weight of $\mathcal{L}_{DDF}$ | 1.5 | 1.5 |
| Total Batch Size | 8 | 8 |
| EMA Decay | 0.9999 | 0.9999 |
| Epochs (w + wo Adv. Aug.) | 100 + 20 | 100 + 20 |
| Number of Candidate Feature | 16 | 4 |
| Rank in iAttn | 4 | 4 |
| Hidden Layer Dimension for $INRLCS(\cdot)$ | [32,] | [32,] |
| Hidden Layer Dimension for $\Phi(\cdot)$ | [256,] | [256,] |
| Using Denser INR Functions | False | True |
| Encoder Attention Type | AIFI | Window Attention |

Table 18: Experiment settings of iDETR and iDETR-DC5 on VisDrone.

| Setting | iDETR | iDETR-DC5 |
|---|---|---|
| Backbone Name | HGNetv2-B4 | HGNetv2-B4 |
| Optimizer | AdamW | AdamW |
| Embedding Dimension | 256 | 256 |
| Feedforward Dimension | 1024 | 1024 |
| Decoder Layers | 6 | 6 |
| Queries | 500 | 500 |
| Sampling Point Number | 12 | 12 |
| Base LR | 2e-4 | 2e-4 |
| Backbone LR | 2e-5 | 2e-5 |
| Weight Decay | 6.5e-5 | 6.5e-5 |
| Weight of $\mathcal{L}_{VFL}$ | 1 | 1 |
| Weight of $\mathcal{L}_{BBox}$ | 5 | 5 |
| Weight of $\mathcal{L}_{GIOU}$ | 2 | 2 |
| Weight of $\mathcal{L}_{FGL}$ | 0.15 | 0.15 |
| Weight of $\mathcal{L}_{DDF}$ | 1.5 | 1.5 |
| Total Batch Size | 8 | 8 |
| EMA Decay | 0.9999 | 0.9999 |
| Epochs (w + wo Adv. Aug.) | 100 + 20 | 100 + 20 |
| Number of Candidate Feature | 16 | 4 |
| Rank in iAttn | 4 | 4 |
| Hidden Layer Dimension for $INRLCS(\cdot)$ | [32,] | [32,] |
| Hidden Layer Dimension for $\Phi(\cdot)$ | [256,] | [256,] |
| Using Denser INR Functions | False | True |
| Encoder Attention Type | AIFI | Window Attention |

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state all core contributions, technical innovations and application scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We analyze the limitations of the proposed method in the experimental section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper focuses on empirical algorithm design and experimental validation, and does not contain theoretical theorems or formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All experimental settings, network architectures and implementation details are elaborated in the main paper and appendix for reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release the anonymized source code, experimental scripts and detailed usage instructions upon publication.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Data splits, hyperparameters, optimizer settings and training strategies are fully provided in the experimental section and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following the standard convention of mainstream object detection works, we do not report error bars. The performance variance of modern object detection models is inherently small across independent runs, making error bars unnecessary for the main comparison.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The hardware configuration, GPU type, memory cost and approximate training time are stated in the experimental section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research strictly follows the NeurIPS Code of Ethics, with no ethical issues or inappropriate applications.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work is fundamental algorithmic research without direct societal impact or specific deployment scenarios.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper uses only public standard datasets and general models with no high risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All adopted public datasets and baseline codes are properly cited, and their open-source licenses are fully respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release an anonymized code repository, which includes documentation for reproduction.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve human subjects research, so IRB approval is not required.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are only used for paper writing and polishing, and are not part of the core method or technical contribution.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.