Dynamic Query as Budgeting for Tiny Object Detection

Yi Zhang[1] **Di Xiong**[1] **Ge Gao**[1] **Xiangyue Zhang**[1]
Yihang Qiu[1] **Yuxuan Zhou**[1] **Shuo Chen**[1,2]*
[1]School of Intelligence Science and Technology, Nanjing University, China
[2]Center for Advanced Intelligence Project, RIKEN, Japan
zhangyi_magic@smail.nju.edu.cn shuo.chen@nju.edu.cn

Abstract

Recent DETR-based tiny object detectors adopt dynamic query mechanisms to handle severe density imbalance. However, existing designs entangle two effects: how many and which queries are decoded. This entanglement obscures what drives the gains and complicates dynamic-query design; additional learned predictors and heuristic filtering also introduce overhead and potential selection errors. We disentangle these effects and show that dynamic query primarily acts as *budgeting*—allocating decoder query capacity according to object density—rather than intricate query selection. Motivated by this view, we formulate a monotonic and conservative budgeting principle and propose a budgeting-based DETR (**BUTR**), a streamlined framework that uses a lightweight, non-learned budgeter to estimate an input-dependent query budget $K(x)$ from encoder confidences. BUTR retains standard Top-$K(x)$ query initialization without additional learned modules or heuristic filtering in the budgeter. At 800×800 resolution, BUTR-L achieves 35.7 AP at 47.8 FPS on AI-TOD-v2 and 29.8 AP at 42.1 FPS on VisDrone on an NVIDIA T4 GPU. Inference-time integration on COCO further demonstrates its applicability to general scenario detection. Code is available at `https://github.com/zhangyi-magic/BUTR`.

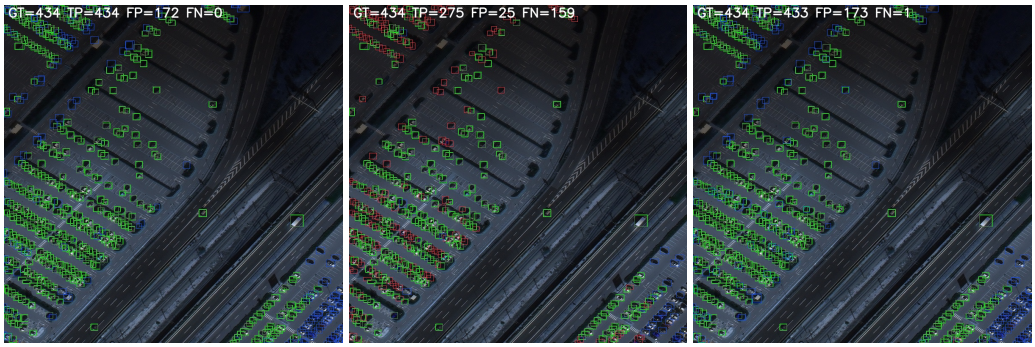
1 Introduction

DETR-based detectors frame object detection as set prediction, simplifying the pipeline by reducing hand-crafted heuristics [1, 2]. Recent advances have improved convergence and efficiency, making this paradigm increasingly practical [3, 4, 5]. Detecting tiny objects, however, remains challenging. Tiny objects occupy only a few pixels, suffer from severe ambiguity, and are easily confused by background clutter. In aerial imagery, this challenge is amplified by density imbalance: most images are sparsely populated, while a tiny fraction contains crowded scenes with hundreds (or more) instances [6, 7, 8, 9]. This creates a tension in DETR design: a fixed decoder query budget is efficient for the majority of images but may be insufficient for crowded cases, whereas uniformly increasing the budget imposes unnecessary overhead and can increase false positive detections in sparse scenes.

To address this, recent works introduce dynamic query mechanisms that adapt decoder capacity to each input. DQ-DETR [10] first predicts a density map and then uses a classifier to assign each image to a preset budget bin. This two-stage discretized design is prone to error propagation, which can lead to unstable budget assignment across density scales. In contrast, Dome-DETR [11] adopts a selection-heavy procedure (Progressive Adaptive Query Initialization, PAQI) that combines a learned window-mask predictor with heuristic filtering, introducing sample-dependent control flow and multiple thresholds. Although effective on challenging dense cases, these methods introduce extra predictors or heuristic operators, increasing complexity and reducing practicality without always

*Corresponding author.

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



(a) Dynamic (b) Fixed Top-300 (c) Fixed Top-600

Figure 1: Qualitative comparison on AI-TOD-v2. Detection outcomes are color-coded as true positives (TP, green), false positives (FP, blue), and false negatives (FN, red). Replacing Dome-DETR's dynamic-query initialization in (a) with a fixed Top-600 initializer in (c) yields highly similar detection results, whereas the smaller fixed budget in (b) (Top-300) is clearly insufficient for this extreme-density scene.

improving accuracy. More importantly, selection-heavy designs entangle query budgeting with query selection, obscuring the true source of improvement.

In this paper, we revisit dynamic query from a mechanism-first perspective. Existing designs conflate two effects: changing how many queries are decoded, and changing which queries survive. To disentangle them, we analyze Dome-DETR [11] through controlled inference-time analysis. This leads to two key observations. First, density-stratified evaluation reveals that, on AI-TOD-v2, PAQI's advantage over a fixed budget sufficient for the majority of images is concentrated on the rare high-density tail and largely obscured by overall AP. Second, a simple confidence-ranked Top-K initializer closely reproduces PAQI's accuracy not only in overall AP but also within density subsets when its fixed budget is adjusted to provide comparable query capacity; in particular, increasing K also closes the gap on the high-density tail. A qualitative example is shown in Fig. 1. Supplementary experiments with DDQ-style initialization [12] further support this interpretation: when the query count is held fixed, replacing confidence-ranked Top-K with spatially distinct query selection yields the same overall AP on both AI-TOD-v2 and VisDrone (App. A). Together, these findings suggest that dynamic query primarily acts as adaptive *budgeting*—determining *how many* queries to decode—rather than intricate selection of *which* queries to keep.

Motivated by this budgeting perspective, we propose a budgeting-based DETR (BUTR), a streamlined framework that realizes dynamic query through a lightweight, non-learned budgeter that estimates a per-image query budget $K(x)$ from a simple density proxy over encoder proposals, while retaining standard Top-$K(x)$ initialization of two-stage DETRs. Compared with prior dynamic-query detectors, BUTR delivers higher accuracy with lower overhead, faster training, and lower T4 deployment latency; as a drop-in modification for modern two-stage DETRs, it also makes dynamic query easier to implement, reproduce, and extend. Our main contributions are summarized as follows:

- **Perspective.** We disentangle how many and which queries are decoded, showing that dynamic query mainly performs adaptive decoder budgeting, with gains concentrated on the rare high density tail revealed by density-stratified evaluation.
- **Method.** We formulate a budgeting principle for dynamic query and instantiate it as BUTR, which estimates a per-image decoder query budget $K(x)$ with a lightweight, non-learned budgeter to enable standard Top-$K(x)$ query initialization, avoiding extra selection machinery.
- **Results.** On AI-TOD-v2 and VisDrone, BUTR achieves higher AP over prior dynamic-query detectors with lower overhead, and markedly faster training and lower T4 deployment latency; additional COCO results further indicate its applicability beyond tiny object detection.

2 Related Work

2.1 Fixed-Query DETRs

DETR [1] formulates detection as end-to-end set prediction with a fixed set of object queries. Subsequent variants have greatly improved convergence and efficiency, including deformable attention [2],

query design and anchor-based query formulations [13, 14, 15], denoising or hybrid matching strategies [16, 17, 18, 19], and recent real-time architectures [3, 4, 5, 12], while retaining a fixed query budget (typically 300–900 queries). On aerial tiny-object benchmarks such as VisDrone [20] and AI-TOD-v2 [6, 7], fixed-query DETR-style detectors, together with related transformer-augmented two-stage baselines such as DNTR [21], achieve strong performance. However, under severe density imbalance, a fixed global query budget creates a natural trade-off: it is efficient for the majority of sparse images, but may become insufficient for rare crowded scenes. This tension motivates recent efforts toward input-adaptive query allocation.

2.2 Dynamic-Query DETRs

To address this tension, recent works introduce dynamic queries that adapt decoder capacity to each input. DQ-DETR [10] adopts a learned strategy by predicting an image-level density signal and selecting among preset budget buckets. Dome-DETR [11], by contrast, follows a selection-style strategy that filters a large candidate set through a multi-stage initialization pipeline (e.g., PAQI), yielding a variable-length decoder input. Related adaptive-query ideas have also appeared in nearby remote-sensing settings, such as FDA-DETR [22] for oriented small-object detection and CGAQ-DETR [23] for synthetic aperture radar object detection. These methods highlight the promise of dynamic query, but also leave open a basic mechanism question: does it help mainly by changing how many queries are decoded, or by changing which queries are retained?

3 Rethinking Dynamic Query: A Budgeting View

We analyze a strong recent dynamic-query method, Dome-DETR, as a diagnostic case, and decompose its query initialization behavior, since its explicit initialization pipeline, PAQI, makes the underlying mechanism directly testable. PAQI starts from the Top-$K_{\max}$ encoder proposals and applies three operators: (i) **SPLIT**, which protects the Top-$K_{\min}$ proposals; (ii) **MASK**, which filters the remaining proposals using a predicted window mask; and (iii) **NMS**, which performs density-adaptive, class-aware suppression, yielding variable-length decoder queries. To isolate the role of each component and avoid retraining-induced confounders, we toggle these operators individually at inference-time.

Table 1: Inference-time ablation on AI-TOD-v2 and VisDrone. A Dome-DETR-L is trained on the respective `trainval` split and evaluated on the `test` set. For each configuration, we report query count $|Q|$ (mean $\mu \pm$ std σ) and overall AP. Δ is relative to the full PAQI pipeline (Row 1).

Init.	Configuration			AI-TOD-v2		VisDrone	
	Split	Mask	NMS	|Q| ($\mu \pm \sigma$)	AP (Δ)	|Q| ($\mu \pm \sigma$)	AP (Δ)
Full PAQI	✓	✓	✓	630 ± 229	35.1	388 ± 21	29.3
Operator Variants	✓	✓	×	875 ± 395	35.1 ($\pm$0.0)	494 ± 14	29.3 ($\pm$0.0)
	✓	×	✓	1083 ± 116	35.1 ($\pm$0.0)	449 ± 14	29.3 ($\pm$0.0)
	×	✓	×	759 ± 484	32.8 ($-$2.3)	491 ± 23	29.2 ($-$0.1)
	×	✓	✓	461 ± 299	32.7 ($-$2.4)	299 ± 40	28.9 ($-$0.4)
	×	×	✓	895 ± 132	35.0 ($-$0.1)	306 ± 36	29.0 ($-$0.3)
Fixed Top-K	×	×	×	100	31.7 ($-$3.4)	100	27.0 ($-$2.3)
				300	34.6 ($-$0.5)	300	29.1 ($-$0.2)
				600	35.0 ($-$0.1)	400	29.3 ($\pm$0.0)
				900	35.1 ($\pm$0.0)	600	29.4 ($+$0.1)
				1500	35.1 ($\pm$0.0)	1200	29.5 ($+$0.2)

From selection to budgeting. Tab. 1 reveals a clear contrast: removing SPLIT causes substantial AP drops (Rows 4–6), whereas removing either MASK or NMS while retaining SPLIT leaves overall AP unchanged despite varying query counts (Rows 1–3). This contrast highlights the key role of SPLIT: preserving the Top-$K_{\min}$ proposals, including high-confidence foreground proposals that the filtering operators may otherwise discard (see App. C for examples). PAQI can therefore be simplified to confidence-ranked proposal retention, naturally degenerating into a plain Top-K initializer.

We then sweep the fixed budget K. The Fixed Top-K rows (Rows 7–11) show that a fixed budget close to PAQI's average query count closely reproduces overall accuracy on both datasets (AI-TOD-v2: AP 35.1 with mean $|Q|$=630 vs. AP 35.0 with fixed K=600; VisDrone: AP 29.3 with mean $|Q|$=388 vs. AP 29.3 with fixed K=400). Here, budget matching refers to approximate matching of

the average query count, not an exact per-image match. This indicates that once the decoder receives a sufficient number of high-confidence proposals, PAQI contributes little to overall AP. App. D provides two complementary explanations: PAQI already has high overlap with the confidence-ranked Top-$|Q|$ proposals, with a mean overlap of $71.5\% \pm 8.1\%$ on AI-TOD-v2 and $84.6\% \pm 1.9\%$ on VisDrone; and encoder proposals exhibit substantial redundancy around foreground objects.

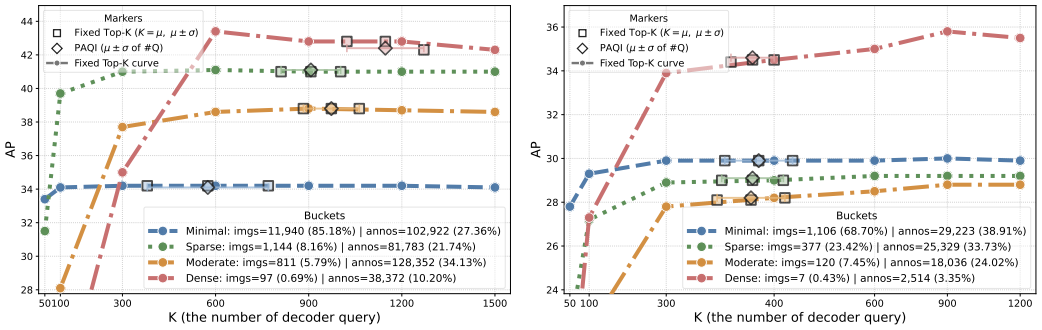
Table 2: Coarse density-stratified evaluation on AI-TOD-v2 and VisDrone.

Init.	AI-TOD-v2				VisDrone			
	`Non-extreme`		`Extreme`		`Non-dense`		`Dense`	
	$\lvert Q\rvert$ ($\mu \pm \sigma$)	AP (Δ)	$\lvert Q\rvert$ ($\mu \pm \sigma$)	AP (Δ)	$\lvert Q\rvert$ ($\mu \pm \sigma$)	AP (Δ)	$\lvert Q\rvert$ ($\mu \pm \sigma$)	AP (Δ)
PAQI	629 ± 227	35.1	1326 ± 100	29.9	388 ± 22	29.1	386 ± 14	34.6
Fixed Top-K	600	35.1 (± 0.0)	600	23.4 (-6.5)	400	29.1 (± 0.0)	400	34.5 (-0.1)
	1500	35.1 (± 0.0)	1500	29.9 (± 0.0)				

Where does dynamic query matter? Tab. 1 shows a clear saturation effect: once a fixed K budget is sufficiently large, further increasing K barely changes the overall AP. This can be misleading under heavy density imbalance: a larger query count is expected to improve recall on dense images, yet this benefit may be diluted in the aggregate metric. We therefore perform a coarse density-stratified analysis using thresholds of #GT=600 on AI-TOD-v2 and #GT=300 on VisDrone, where GT denotes ground-truth objects. These thresholds are chosen near the observed saturation points with acceptable annotation counts, yielding `non-extreme`/`extreme` and `non-dense`/`dense` splits, respectively. On AI-TOD-v2, this gives 13,992 `non-extreme` images (99.81%) with 351,429 annotations (93.44%) versus 26 `extreme` images (0.19%) with 24,692 annotations (6.56%); on VisDrone, it gives 1,603 `non-dense` images (99.57%) with 72,588 annotations (96.66%) versus 7 `dense` images (0.43%) with 2,514 annotations (3.34%).

As shown in Tab. 2, on AI-TOD-v2 PAQI behaves like Top-600 on the `non-extreme` majority, while its gains are confined to the rare `extreme` tail, where it increases the budget to 1326 ± 100; once the fixed budget is raised accordingly, Top-1500 closes the gap. In contrast, VisDrone has a much weaker tail (max #GT=461): PAQI's query count barely changes across the split (388 ± 22 vs. 386 ± 14), and a budget-matched fixed baseline (Top-400) reproduces both subsets. Thus, the benefit of PAQI is confined to the high-density tail and is fully reproducible with a matched fixed budget.

Budget matching reproduces density-wise accuracy. The finer bucket-wise AP–K curves in Fig. 2 further support this interpretation. Across density buckets, PAQI consistently lies near the response curve of a budget-matched Top-K baseline, indicating that its behavior is largely explained by adaptive budgeting. Moreover, AP quickly enters a saturation regime as K increases, while PAQI often allocates more queries than needed, incurring extra compute with little or no accuracy gain. This broad saturation regime explains why a simple density proxy suffices for dynamic budgeting: moderate estimation error rarely moves the operating point out of the flat region.



(a) AI-TOD-v2 (b) VisDrone

Figure 2: Fine density-stratified AP–budget response. Curves show AP versus fixed decoder budget K within each density bucket: *Minimal* (#GT ≤ 50), *Sparse* ($50 <$ #GT ≤ 100), *Moderate* ($100 <$ #GT ≤ 300), and *Dense* ($300 <$ #GT ≤ 600). Diamonds mark PAQI's bucket-wise mean query count μ with horizontal bars for $\pm\sigma$; squares denote budget-matched fixed Top-K baselines at $K \in \{\mu - \sigma, \mu, \mu + \sigma\}$.

Further evidence for the budgeting interpretation. The preceding analyses suggest that the primary role of dynamic query is to adapt decoder capacity to the object density of each image rather

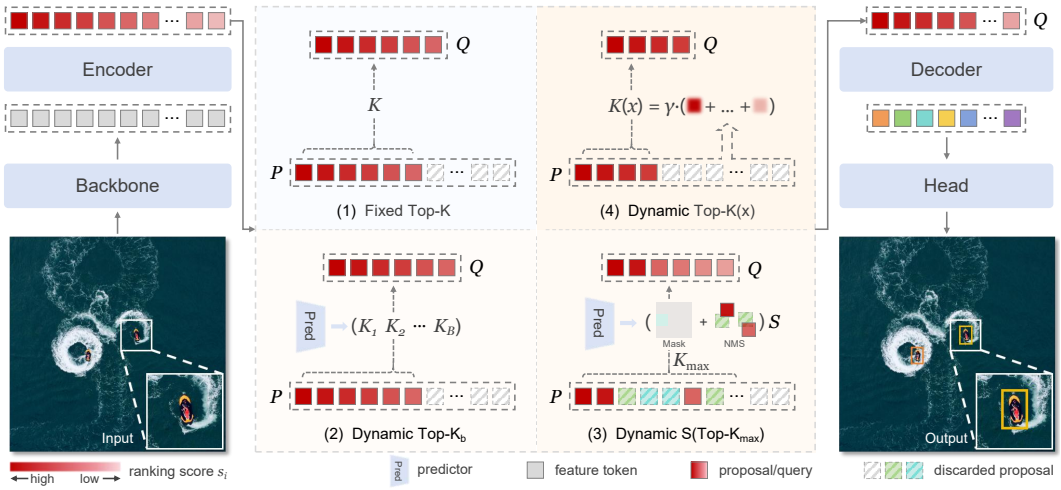


Figure 3: Overview of query initialization: (1) fixed-query DETRs (e.g., Deformable DETR [2], DINO [17], and RT-DETR [3]); dynamic-query DETRs include (2) DQ-DETR [10], (3) Dome-DETR [11], and (4) BUTR.

than to perform intricate query selection. If this interpretation holds, changing *which* queries are retained while keeping *how many* fixed should yield nearly unchanged performance. We test this prediction using DDQ-style initialization [12], which retains a fixed-size, spatially distinct subset of candidates. We separately train D-FINE-L with DDQ-style and original Top-K initialization, using each initializer during both training and inference at the same query count; the two variants achieve the same overall AP on both AI-TOD-v2 and VisDrone. App. A provides the motivation, rationale for choosing D-FINE, and detailed results. Together with the PAQI interventions, these results support a mechanism-first interpretation: adaptive decoder budgeting accounts for much of the observed benefit of dynamic query in the evaluated settings, motivating BUTR's minimal design.

4 BUTR

4.1 Preliminaries

A unified view of query initialization. For clarity, we summarize query initialization in two-stage DETRs under a unified notation, as illustrated in Fig. 3. Given an input image x, let $P = \{(\boldsymbol{z}_i, \boldsymbol{r}_i, \boldsymbol{\ell}_i)\}_{i=1}^{L}$ denote the set of encoder proposals, where $\boldsymbol{z}_i \in \mathbb{R}^D$ is the i-th proposal feature (or learned embedding), $\boldsymbol{r}_i$ is its reference box, and $\boldsymbol{\ell}_i = (\ell_{i1}, \ldots, \ell_{iC}) \in \mathbb{R}^C$ is its classification logit vector, where ℓ_{ic} is the logit for class c. Let $\mathcal{T}_K(\cdot)$ denote the Top-K truncation operator based on a proposal ranking score $s_i = \phi(\boldsymbol{\ell}_i)$. Standard two-stage DETRs (e.g., Deformable DETR [2], DINO [17], and RT-DETR [3]) use a fixed budget: $Q = \mathcal{T}_K(P)$ with constant K. By contrast, dynamic-query variants adopt different strategies: DQ-DETR [10] performs discrete learned binning, $Q = \mathcal{T}_{K_b}(P)$ where $K_b \in \{K_1, \ldots, K_B\}$; selection-style methods such as Dome-DETR [11] apply an additional selection pipeline to an enlarged candidate pool, $Q = \mathcal{S}\big(\mathcal{T}_{K_{\max}}(P)\big)$, with $\mathcal{S}(\cdot)$ denoting the selection pipeline. Our BUTR instead keeps the standard Top-K initialization form and only makes the truncation point adaptive based on proposal confidences: $Q = \mathcal{T}_{K(x)}(P)$, where $K(x)$ is a per-image budget. This design introduces neither learned discrete predictors for K_b nor a post-Top-K selection pipeline $\mathcal{S}$, making BUTR a drop-in modification for two-stage DETRs.

4.2 Budgeter: Estimating an Adaptive Query Budget

Budgeting principle. Following the diagnosis in Sec. 3, we identify that the essential role of dynamic query is to determine *how many* decoder queries are needed for a given input. Formally, we view a budgeter as a policy $\mathcal{B} : \mathcal{X} \to \mathcal{K}$ that assigns each input image $\boldsymbol{x} \in \mathcal{X}$ a feasible decoder budget. A reasonable budgeting policy should satisfy two properties: (i) *monotonicity*—denser images should not receive fewer queries; and (ii) *conservativeness*—the policy should be mildly biased toward sufficient capacity, as under-allocation is more harmful than slight over-allocation in the observed AP–budget saturation regime (Fig. 2). Under this view, budgeting does not require recovering the

exact object count, but only preserving a reliable ordering of image density and translating this ordering into an appropriately conservative query budget.

In BUTR, we provide a minimal implementation through a factorization $\mathcal{B} = g \circ e$, where $e : \mathcal{X} \to \mathbb{R}_+, g : \mathbb{R}_+ \to \mathcal{K}$. Here, the estimator e maps an input image to a nonnegative density-related signal, while the mapping g converts this signal into a feasible decoder budget. Under this decomposition, the estimator is responsible for preserving density ordering, while the mapping determines how conservatively this ordering is translated into decoder capacity.

Estimator: a density proxy from encoder confidences. While the true object count is unavailable in practice, the estimator e should preserve density ordering of inputs. Fortunately, the ranking score $s_i = \phi(\boldsymbol{\ell}_i)$ already provides a natural and cost-free signal: it quantifies how likely the i-th encoder proposal carries foreground evidence. Intuitively, if more proposals exhibit high scores, the image is likely to be denser and thus require more decoder queries. We therefore define the density proxy as the sum of all ranking scores:

$$e(x) = \sum_{i=1}^{L} s_i = \sum_{i=1}^{L} \phi(\boldsymbol{\ell}_i). \tag{1}$$

This aggregation naturally respects the desired monotonicity. In this work, we instantiate ϕ with a simple yet effective choice:

$$\phi(\boldsymbol{\ell}_i) = \max_{c \in \{1,\dots,C\}} \sigma(\ell_{ic}) \in [0, 1], \tag{2}$$

where $\sigma(u) = \frac{1}{1+e^{-u}}$ is the sigmoid function. That is, $\phi(\boldsymbol{\ell}_i)$ is the maximum sigmoid-activated class confidence across all classes, which we use as the proposal-level foreground confidence. This choice is compatible with established detection practice, where confidence or quality-aware scores serve as generic indicators of foreground evidence in proposal generation and dense detection [24, 25, 26, 27, 28, 29]. Other instantiations of ϕ are possible and are examined in the ablation study (Sec. 5.4).

Mapping: from density proxy to decoder budget. Given the estimator output $e(x)$, budgeting requires a mapping g that converts this density proxy into a decoder budget. This mapping should naturally be monotone and also encode conservativeness. Other monotone choices are also possible, such as discrete binning functions; in this work, we adopt a simple linear scaling. Specifically, we instantiate the mapping as $g(t) = \gamma t, \gamma \geq 1$, yielding the budget:

$$K(x) = \gamma \cdot e(x). \tag{3}$$

Here, γ is a safety factor controlling how conservatively the density signal is translated into budget. Alg. 1 in App. B summarizes the complete dynamic-query initialization procedure of BUTR.

Empirical properties of the estimator. Empirically, $e(x)$ correlates strongly with the true object count (AI-TOD-v2: Pearson $r = 0.94$, Spearman $\rho = 0.82$; VisDrone: $r = 0.82$, $\rho = 0.79$; all p-values $< 10^{-5}$), supporting its use as a density-related budgeting signal on these datasets.

To summarize, BUTR provides a simple, non-learned budgeter with negligible overhead: it introduces no learnable parameters, auxiliary supervision, additional loss, or budgeter-specific optimization. The host detector is still trained using its detection objectives. By preserving the standard two-stage initialization form, it remains fully compatible with existing two-stage DETR pipelines and can even serve as a plug-in modification without retraining.

5 Experiments

5.1 Setup

Datasets and metrics. Our main comparisons use AI-TOD-v2 [6, 7] and VisDrone [20], two widely used tiny-object benchmarks with stronger density variation than COCO [30]. Sec. 5.5 additionally evaluates COCO to assess applicability beyond tiny-object detection. We report AP, cost, training time, deployment latency, and the average number of decoder queries ($|Q|$), and additionally adopt density-stratified evaluation to expose high-density-tail behavior.

Fair comparison protocol. For the main comparisons, we set the optional feasible range $[K_{\min}, K_{\max}]$ in Alg. 1 to match Dome-DETR's [11] decoder query budget ranges: $[300, 1500]$ on AI-TOD-v2 and $[250, 500]$ on VisDrone. The common upper bound rules out a larger permitted

maximum query budget as the source of accuracy gains, while the common lower bound rules out a smaller permitted minimum query budget as the source of efficiency gains. These ranges serve the comparison protocol rather than define the budget estimator; Fig. 4 separately examines its behavior without dataset-specific clipping. DQ-DETR [10] is reported under a 36-epoch schedule on both datasets. For fixed-query baselines, we adapt them to the tiny and dense regime by using 600/400 queries on AI-TOD-v2/VisDrone, following the saturation points diagnosed in Sec. 3, and by extending training to 160/120 epochs to match Dome-DETR and BUTR, with epoch-based schedules scaled proportionally (e.g., data-augmentation milestones). Optimization and objective settings (e.g., learning rate and loss-related hyperparameters) are aligned with each dataset; for DEIM, we replace its Matchability-Aware Loss (MAL) [5] with Varifocal Loss [29], since MAL is unstable in our setting and leads to collapsed AP. For these tiny-object comparisons, BUTR, D-FINE, DEIM, and Dome-DETR use four feature levels with an anchor size of 0.0125; DQ-DETR retains its five-level architecture. Models are trained on the `trainval` split and evaluated on the `test` split. Following Dome-DETR, the L-scale configurations use four decoder layers on AI-TOD-v2 and six on VisDrone, accounting for their different parameter counts.

Implementation details. Training uses a server with two 7R32 CPUs and eight RTX 4090 GPUs, with 800×800 inputs. Training time is measured with total batch size 16 and per-epoch validation. Deployment latency is measured on a single NVIDIA Tesla T4 (16 GB) with TensorRT 10.4.0, FP16 enabled, and batch size 1; precision constraints and the measurement protocol are detailed in App. H.

5.2 Comparisons

Table 3: Comparison on AI-TOD-v2. DQ-DETR results use the official checkpoint; others are our re-implementations. Dynamic-query GFLOPs and $|Q|$ are averages. Tr. (h): training time (hours); Lat. (ms): T4 TensorRT latency (milliseconds). Ratios are computed relative to Dome-DETR at the same model scale.

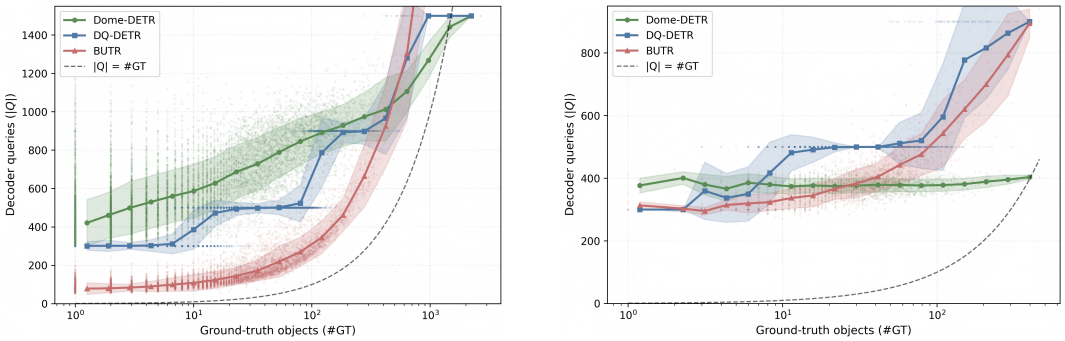
Method	#Params	GFLOPs	$\lvert Q\rvert$	Tr. (h)	Lat. (ms)	AP	AP_{50}	AP_{75}	AP_{vt}	AP_t	AP_s	AP_m
Fixed-query detectors												
D-FINE-S [4]	11.6M	115.1	600	33.7	11.27	32.1	65.5	27.4	17.5	32.3	38.1	45.7
D-FINE-M [4]	22.3M	248.2	600	40.4	16.86	34.3	68.0	30.3	19.7	34.8	39.1	47.7
D-FINE-L [4]	31.9M	329.6	600	43.0	20.34	35.2	69.3	31.1	20.4	36.0	40.0	47.9
DEIM-S [5]	11.6M	115.1	600	36.6	11.26	29.0	61.0	23.9	16.8	28.6	35.0	41.7
DEIM-M [5]	22.3M	248.2	600	43.4	16.92	33.7	67.3	30.1	20.3	34.3	38.4	46.1
DEIM-L [5]	31.9M	329.6	600	46.6	20.34	34.5	68.5	30.7	20.4	35.0	38.9	46.5
Dynamic-query detectors												
DQ-DETR [10]	58.7M	1805.4	410	14.8	205.58	30.5	69.2	22.7	15.2	30.9	36.8	45.5
Dome-DETR-S [11]	13.2M	165.3	639	65.0	23.77	32.3	67.0	27.5	18.2	32.4	38.7	45.6
Dome-DETR-M [11]	23.9M	298.5	632	78.9	30.01	34.0	68.4	29.8	19.6	34.3	39.5	48.3
Dome-DETR-L [11]	33.4M	379.8	630	83.6	33.30	35.1	69.6	30.8	19.2	35.6	39.9	47.7
BUTR-S (Ours)	11.6M	112.5	319	30.8 (2.1×)	12.23 (1.9×)	**32.6** (+0.3)	67.2	27.8	19.0	32.2	38.6	45.4
BUTR-M (Ours)	22.3M	244.8	317	37.2 (2.1×)	17.60 (1.7×)	**34.6** (+0.6)	69.2	30.5	20.3	34.7	38.9	47.2
BUTR-L (Ours)	31.9M	326.1	316	39.8 (2.1×)	20.94 (1.6×)	**35.7** (+0.6)	69.7	32.2	21.2	36.3	40.6	49.3

AI-TOD-v2. Tab. 3 shows that a strong fixed-query baseline, D-FINE, already surpasses Dome-DETR in overall AP. One plausible reason is that Dome-DETR mistakenly discards high-confidence proposals aligned with foreground objects; App. C provides qualitative examples. This result further supports our budgeting view. Compared with fixed-query baselines, BUTR retains the same model size and adds 0.6–1.0 ms of T4 deployment latency relative to D-FINE, while achieving higher training efficiency and better overall AP across scales; for example, BUTR-L reaches 35.7 AP versus 35.2 for D-FINE-L, with about half as many queries on average (316 vs. 600) and slightly lower computation (326.1 vs. 329.6 GFLOPs). Compared with dynamic-query detectors, BUTR is both more accurate and substantially more efficient; for example, BUTR achieves higher AP with fewer parameters and lower computation, while requiring about $2.1\times$ less training time and delivering 1.6–$1.9\times$ faster T4 inference than same-scale Dome-DETR.

VisDrone. Compared with AI-TOD-v2, VisDrone has a milder density tail (max $\#\mathrm{GT}{=}461$, with only one image exceeding 400 objects), so the gains from dynamic query are expected to be smaller under our budgeting view. Tab. 4 supports this expectation. In this regime, BUTR achieves higher accuracy than Dome-DETR with 1.5–$1.8\times$ faster T4 inference. Relative to D-FINE, it achieves a slight AP gain with similar GFLOPs and 1.7–2.0 ms of additional deployment latency.

Table 4: Comparison on VisDrone. Results are from our re-implementations. Dynamic-query GFLOPs and $|Q|$ are averages. Tr. (h): training time (hours); Lat. (ms): T4 TensorRT latency (milliseconds).

Method	#Params	GFLOPs	$\|Q\|$	Tr. (h)	Lat. (ms)	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Fixed-query detectors											
D-FINE-S [4]	11.6M	113.3	400	6.7	11.39	26.6	45.1	27.0	16.8	37.0	53.6
D-FINE-M [4]	22.3M	245.9	400	9.2	17.35	29.2	48.9	30.0	18.9	40.5	53.8
D-FINE-L [4]	34.5M	329.5	400	11.9	22.06	29.6	49.6	30.2	19.0	40.8	55.7
DEIM-S [5]	11.6M	113.3	400	8.5	11.23	21.9	41.1	22.4	17.3	27.5	31.7
DEIM-M [5]	22.3M	245.9	400	10.7	17.40	24.5	44.6	24.9	18.4	31.8	36.8
DEIM-L [5]	34.5M	329.5	400	13.4	22.37	25.2	45.4	25.2	18.4	32.3	38.2
Dynamic-query detectors											
DQ-DETR [10]	58.7M	1782.4	512	4.5	211.18	18.4	33.7	17.9	12.0	27.0	35.8
Dome-DETR-S [11]	13.2M	162.9	389	14.0	23.25	26.4	45.1	26.8	16.4	36.9	51.0
Dome-DETR-M [11]	23.9M	295.4	384	19.9	29.69	28.9	48.5	29.5	18.8	40.0	55.2
Dome-DETR-L [11]	36.0M	379.0	388	22.4	34.47	29.3	49.2	30.0	18.6	40.8	56.4
BUTR-S (Ours)	11.6M	114.9	416	6.8 (2.1×)	13.26 (1.8×)	**26.7** (+0.3)	45.2	27.4	16.8	37.3	52.5
BUTR-M (Ours)	22.3M	245.8	396	9.3 (2.1×)	19.31 (1.5×)	**29.4** (+0.5)	49.4	29.9	18.9	40.7	57.0
BUTR-L (Ours)	34.5M	329.5	398	11.8 (1.9×)	23.76 (1.5×)	**29.8** (+0.5)	50.1	30.4	19.1	41.1	57.1



(a) AI-TOD-v2 (b) VisDrone

Figure 4: Query usage versus image density. Each point corresponds to one test image; solid lines and shaded regions denote the binned mean and standard deviation, respectively. The dashed gray line is an ideal reference proportional to #GT. BUTR shows the clearest monotonic increase with image density.

Query usage behavior. Fig. 4 provides a direct view of the budgeting behavior of dynamic-query methods. We disable the dataset-specific clipping to observe BUTR's native budget estimates. BUTR exhibits a clear increasing trend in allocated queries with #GT, whereas DQ-DETR is limited by discrete budget bins and Dome-DETR shows weaker alignment with density.

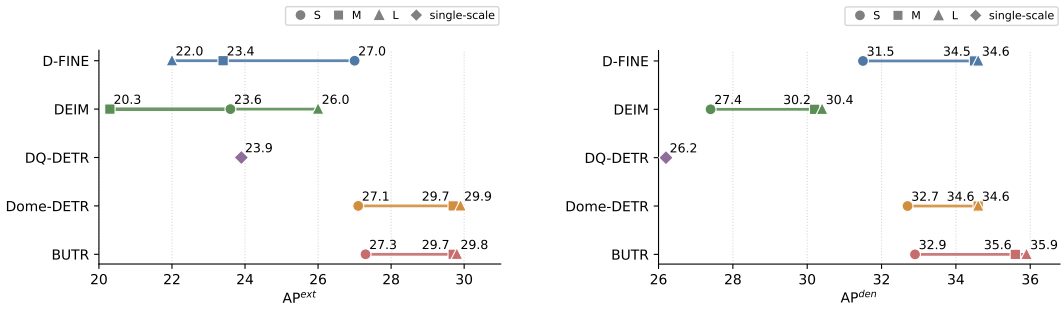
5.3 Density-Stratified Evaluation

Overall AP can obscure performance on the rare high-density tail. Following Sec. 3, we examine the `extreme` subset of AI-TOD-v2 (AP^{ext}) and the `dense` subset of VisDrone (AP^{den}), where insufficient decoder capacity is particularly consequential. Fig. 5a shows that the benefits of dynamic query on AI-TOD-v2 are more pronounced on the tail subset than in overall AP. Increasing model scale, however, does not consistently improve tail performance for fixed-query detectors: D-FINE achieves 27.0, 23.4, and 22.0 AP^{ext} at the S, M, and L scales, respectively. BUTR remains competitive with Dome-DETR on AP^{ext} and achieves higher AP^{den} across scales (Fig. 5b). Its gains also extend beyond the tail, as the full results in App. E show improvements across multiple density strata.

Because the tail subsets contain few images, we examine AP^{ext} stability for the L-scale models on AI-TOD-v2. Under both repeated seed-0 runs and runs with matched seeds 0, 1, and 2, BUTR-L exhibits lower AP^{ext} variability than Dome-DETR-L. App. F provides the protocols and results.

5.4 Ablations

Tab. 5 ablates two aspects of BUTR-L on AI-TOD-v2: the estimator e and the mapping g. For the estimator, all variants are retrained and adopt our budget rule (Eq. 3) with $\gamma = 1.0$. Specifically,



(a) AI-TOD-v2: `extreme` subset

(b) VisDrone: `dense` subset

Figure 5: Density-stratified tail performance across model families and scales. (a) AP^{ext} on the `extreme` subset of AI-TOD-v2. (b) AP^{den} on the `dense` subset of VisDrone.

we compare four instantiations of the ranking function ϕ that defines $s_i = \phi(\boldsymbol{\ell}_i)$: *Max* with $s_i = \max(0, \max_c \boldsymbol{\ell}_{ic})$, *Max-sigmoid* with $s_i = \max_c \sigma(\boldsymbol{\ell}_{ic})$, *Sum-prob* with $s_i = \sum_c \sigma(\boldsymbol{\ell}_{ic})$, and *Union-prob* with $s_i = 1 - \prod_c (1 - \sigma(\boldsymbol{\ell}_{ic}))$. The results show that BUTR is not sensitive to the exact choice of ϕ. Among them, we choose the *Max-sigmoid* proxy as the default setting. We then ablate the budget mapping using *Max-sigmoid* estimator, considering two variants: (i) varying the safety factor γ in Eq. 3, and (ii) a discrete binning strategy that quantizes the soft density proxy $e(x)$ into one of a set of predefined budget levels. Specifically, we adopt the budget bin setting $\{300, 500, 900, 1500\}$ from DQ-DETR [10] as the discrete targets. We find that the model is robust to γ over a wide range (1.0–3.0). The discrete binning baseline achieves comparable accuracy but requires manually designed thresholds. We adopt $\gamma = 2.0$ as the default setting.

Table 5: Ablations of budget estimation and mapping methods for BUTR-L on AI-TOD-v2. Estimator e variants are trained and evaluated without an additional mapping (i.e., $\gamma = 1.0$), while the mapping g study is conducted by inference-time ablation using the *Max-sigmoid* estimator. Tr. (h): training wall-clock time.

Category	Variant	$\lvert Q \rvert$	Tr. (h)	AP	AP_{50}	AP_{75}	AP_{vt}	AP_{t}	AP_{s}	AP_{m}	AP^{ext}
Estimator e	*Max*	302	37.8	35.6	69.6	31.5	20.4	36.4	40.4	49.0	27.7
	Max-sigmoid	303	37.8	35.5	69.3	32.2	21.1	36.1	40.5	49.1	28.9
	Sum-prob	304	41.5	35.3	69.4	31.1	20.3	36.0	39.6	47.0	31.0
	Union-prob	304	40.7	35.2	68.9	31.4	18.9	36.0	40.2	48.5	28.3
Mapping g	$\gamma = 1.2$	305	/	35.6	69.4	32.2	21.2	36.2	40.6	49.1	29.5
	$\gamma = 1.5$	308	/	35.6	69.5	32.2	21.2	36.2	40.6	49.2	29.6
	$\gamma = 2.0$	316	/	35.7	69.6	32.3	21.2	36.2	40.7	49.2	29.8
	$\gamma = 3.0$	340	/	35.7	69.6	32.3	21.2	36.3	40.7	49.3	29.8
	binning	305	/	35.6	69.6	32.2	21.2	36.2	40.6	49.2	29.5

5.5 Generality beyond Tiny Object Scenarios

We first extend the evaluation to dense agricultural imagery using Sorghum-Grain-Count [31], training all compared models on the cropped agricultural dataset as in our AI-TOD-v2 and VisDrone evaluations. BUTR-L matches or exceeds the overall AP of fixed-query baselines using 1100 queries while decoding only 378 queries on average; the full protocol and results are provided in App. G.

We then evaluate BUTR on COCO [30] using D-FINE [4] and DEIM [5] under an *inference-time plug-in* protocol. Unlike the preceding evaluations, we keep the officially provided checkpoints fixed and replace only the decoder query initialization with our budgeter, without retraining. This paired comparison isolates the effect of query budgeting from training-run variability, allowing small AP differences to be examined with identical model weights. As shown in Tab. 6, BUTR reduces the average number of decoder queries and GFLOPs across all variants while maintaining or slightly improving overall AP (by 0.1–0.2 points). We also report paired T4 TensorRT latency for the fixed-query and BUTR-augmented variants under the protocol in App. H.

Taken together, BUTR improves accuracy and efficiency on AI-TOD-v2, which exhibits the heavier density imbalance and more extreme high-density tail of the two aerial benchmarks; its benefits remain evident on the milder VisDrone benchmark and extend to dense agricultural imagery; and even on conventional COCO, it reduces decoder computation without sacrificing overall AP or requiring

Table 6: Generality of BUTR on COCO under an inference-time plug-in setting without retraining. We report average GFLOPs and $|Q|$ for the BUTR-augmented variants. Lat. (ms): T4 TensorRT latency (App. H).

Method	#Params	GFLOPs	$\lvert Q\rvert$	Lat. (ms)	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{s}	AP^{val}_{m}	AP^{val}_{l}
D-FINE-S [4]	10.2M	25.2	300	3.63	48.5	65.4	52.6	29.3	52.2	65.4
D-FINE-S+BUTR	10.2M	24.6	232	4.24	**48.6**(+0.1)	65.5	52.8	29.3	52.1	65.2
D-FINE-M [4]	19.3M	56.7	300	5.66	52.3	69.9	56.4	33.2	56.5	70.2
D-FINE-M+BUTR	19.3M	56.6	289	6.44	**52.4**(+0.1)	70.0	56.5	33.2	56.6	70.1
D-FINE-L [4]	30.8M	91.0	300	7.94	54.0	71.6	58.4	36.5	58.0	71.9
D-FINE-L+BUTR	30.8M	90.2	253	8.81	**54.1**(+0.1)	71.8	58.4	36.6	58.0	72.0
D-FINE-X [4]	61.6M	202.5	300	13.52	55.8	73.7	60.2	37.3	60.5	73.5
D-FINE-X+BUTR	61.6M	201.7	253	13.84	**55.9**(+0.1)	73.9	60.4	37.4	60.6	73.4
DEIM-S [5]	10.2M	25.2	300	3.75	**49.0**	65.9	53.1	30.4	52.6	65.7
DEIM-S+BUTR	10.2M	24.6	239	4.32	**49.0**(±0.0)	66.0	53.2	30.2	52.6	65.5
DEIM-M [5]	19.3M	56.7	300	5.73	52.7	70.0	57.3	35.2	56.7	69.4
DEIM-M+BUTR	19.3M	55.9	236	6.38	**52.8**(+0.1)	70.1	57.4	35.1	56.7	69.5
DEIM-L [5]	30.8M	91.1	300	8.07	54.7	72.4	59.4	36.9	59.6	71.8
DEIM-L+BUTR	30.8M	89.8	224	8.83	**54.9**(+0.2)	72.6	59.6	37.1	59.8	71.8
DEIM-X [5]	61.7M	202.7	300	13.78	56.5	74.0	61.5	38.8	61.4	74.2
DEIM-X+BUTR	61.7M	200.6	183	13.71	**56.6**(+0.1)	74.2	61.7	38.9	61.4	74.0

retraining. These results highlight two complementary advantages of BUTR: compared with existing dynamic-query designs, it avoids the overhead and potential decision errors introduced by additional learned predictors and heuristic filtering in query initialization; compared with fixed-query detectors, it adapts decoder capacity to address the risks of insufficient recall in crowded images under a small budget and increased false positives in sparse images during deployment under a large budget. This combination supports efficient and effective detection across the four datasets evaluated in this work. More broadly, because BUTR builds on proposal confidences and confidence-ranked initialization already present in two-stage detectors, we expect its budgeting approach to be applicable to a broader range of detection scenarios.

6 Conclusion

We revisited dynamic query for tiny object detection from a mechanism-first perspective, finding that adaptive budgeting largely explains Dome-DETR's accuracy in the evaluated settings. Motivated by this view, we proposed BUTR, a principled dynamic query framework that estimates a per-image decoder budget from encoder confidences while retaining standard Top-K initialization form. Experiments on AI-TOD-v2 and VisDrone show improved overall accuracy and efficiency over Dome-DETR, while inference-only integration on COCO reduces queries and computation without degrading overall AP. We hope this perspective informs the future design and evaluation of detectors.

Limitations. BUTR relies on encoder confidences as its budgeting signal. Unlike fixed Top-K selection, which depends on score ranking, score aggregation also depends on their numerical scale; changes in score calibration, proposal count, or domain may therefore affect allocation even when rankings remain unchanged. Our experiments support applicability in the tested domains, while other dense settings, such as crowded products and cell or particle imagery, remain to be evaluated. Dynamic-budget inference also incurs synchronization and engine-boundary overhead in our current TensorRT deployment, so fewer queries do not necessarily translate into lower latency. Finally, the high-density subsets contain few images (26 on AI-TOD-v2 and 7 on VisDrone), so their AP is sensitive to test-sample composition. Density-stratified evaluation should therefore complement, rather than replace, overall evaluation.

Acknowledgments

This work was supported by the New Generation Artificial Intelligence-National Science and Technology Major Project (Nos. 2025ZD0123502), National Natural Science Fund of China (Nos. 62506155), Provincial Natural Science Fund of Jiangsu (Nos. BK20251985), Suzhou Municipal Leading Talents Fund (Nos. ZXL2025320), ZTE Industry-University Project (Nos. IA20260910004), Alipay Industry-University Project (Nos. 2026101543758), and Key Project of the State Key Laboratory for Novel Software Technology (Nos. ZZKT2026A14).

References

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

[2] 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] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16965–16974, 2024.

[4] Yansong Peng, Hebei Li, Peixi Wu, Yueyi Zhang, Xiaoyan Sun, and Feng Wu. D-fine: Redefine regression task in detrs as fine-grained distribution refinement. *arXiv preprint arXiv:2410.13842*, 2024.

[5] Shihua Huang, Zhichao Lu, Xiaodong Cun, Yongjun Yu, Xiao Zhou, and Xi Shen. Deim: Detr with improved matching for fast convergence. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15162–15171, 2025.

[6] Jinwang Wang, Wen Yang, Haowen Guo, Ruixiang Zhang, and Gui-Song Xia. Tiny object detection in aerial images. In *2020 25th international conference on pattern recognition (ICPR)*, pages 3791–3798. IEEE, 2021.

[7] Chang Xu, Jinwang Wang, Wen Yang, Huai Yu, Lei Yu, and Gui-Song Xia. Detecting tiny objects in aerial images: A normalized wasserstein distance and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 190:79–93, 2022.

[8] Chenhongyi Yang, Zehao Huang, and Naiyan Wang. Querydet: Cascaded sparse query for accelerating high-resolution small object detection. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 13668–13677, 2022.

[9] Gong Cheng, Xiang Yuan, Xiwen Yao, Kebing Yan, Qinghua Zeng, Xingxing Xie, and Junwei Han. Towards large-scale small object detection: Survey and benchmarks. *IEEE transactions on pattern analysis and machine intelligence*, 45(11):13467–13488, 2023.

[10] Yi-Xin Huang, Hou-I Liu, Hong-Han Shuai, and Wen-Huang Cheng. Dq-detr: Detr with dynamic query for tiny object detection. In *European Conference on Computer Vision*, pages 290–305. Springer, 2024.

[11] Zhangchi Hu, Peixi Wu, Jie Chen, Huyue Zhu, Yijun Wang, Yansong Peng, Hebei Li, and Xiaoyan 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*, pages 101–110, 2025.

[12] Shilong Zhang, Xinjiang Wang, Jiaqi Wang, Jiangmiao Pang, Chengqi Lyu, Wenwei Zhang, Ping Luo, and Kai Chen. Dense distinct query for end-to-end object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7329–7338, 2023.

[13] 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.

[14] 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*, volume 36, pages 2567–2575, 2022.

[15] 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.

[16] 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.

[17] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

[18] Qiang Chen, Xiaokang Chen, Jian Wang, Shan Zhang, Kun Yao, Haocheng Feng, Junyu Han, Errui Ding, Gang Zeng, and Jingdong Wang. Group detr: Fast detr training with group-wise one-to-many assignment. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6633–6642, 2023.

[19] Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. Detrs with hybrid matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19702–19712, 2023.

[20] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7380–7399, 2021.

[21] Hou-I Liu, Yu-Wen Tseng, Kai-Cheng Chang, Pin-Jyun Wang, Hong-Han Shuai, and Wen-Huang Cheng. A denoising fpn with transformer r-cnn for tiny object detection. *IEEE transactions on geoscience and remote sensing*, 62:1–15, 2024.

[22] Cheng Ju, Yu Zhao, Shuiqing Miao, Dina Li, Rongjun Chai, Yuansha Xie, and Wenyao Yan. Fda-detr: A frequency-aware detr with dynamic query and adaptive multi-task optimization for oriented small object detection. *Plos one*, 20(8):e0330929, 2025.

[23] Zhen Zuo, Zhangjunjie Cheng, Siyang Huang, Junyu Wei, and Zhuoyuan Wu. Cgaq-detr: Detr with corner guided and adaptive query for sar object detection. *Remote Sensing*, 17(18):3254, 2025.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[25] 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.

[26] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: A simple and strong anchor-free object detector. *IEEE transactions on pattern analysis and machine intelligence*, 44(4):1922–1933, 2020.

[27] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9759–9768, 2020.

[28] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Advances in neural information processing systems*, 33:21002–21012, 2020.

[29] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko Sunderhauf. Varifocalnet: An iou-aware dense object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8514–8523, 2021.

[30] 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.

[31] A. Bakshi, S. Pramanik, D. Caragea, I. Somayanda, R. Kumar, M. Bongari, S. R. Bean, Y. Emendack, C. M. Hayes, and K. Jagadish. Sorghum-Grain-Count: A large image dataset and benchmark for sorghum grain count estimation. *Smart Agricultural Technology*, 12:101218, 2025.

A Supplementary Comparison with DDQ-Style Initialization

Why examine DDQ-style initialization? The rationale for choosing DDQ is straightforward: if our PAQI-derived conclusion holds, replacing confidence-ranked Top-K with an initializer that changes *which* queries are retained while keeping *how many* queries fixed should yield nearly unchanged performance. DDQ [12] provides such a mechanism: its distinct-query selection takes a dense set of query candidates and retains a fixed-size, spatially distinct subset. For each dataset, we separately train and evaluate models with DDQ-style initialization and the original Top-K initialization at the same query count, using each initializer during both training and inference. We emphasize that the full DDQ framework also includes inter-layer query selection and corresponding loss designs; here, we adopt only its distinct query initialization mechanism.

Why use D-FINE? Removing PAQI and its associated components from Dome-DETR leaves an architecture close to D-FINE [4]. D-FINE therefore provides a clean base that directly connects this comparison to our PAQI diagnosis: we replace its original Top-K initializer with DDQ-style selection while keeping the query count fixed.

Result and interpretation. As shown in Tab. 7, the separately trained variants achieve the same overall AP: 35.2 on AI-TOD-v2 and 29.6 on VisDrone, using fixed budgets of 600 and 400 queries, respectively. Thus, even when the detector is trained with the alternative initializer, DDQ-style selection yields no observed overall AP advantage over the original initializer in this comparison. This complements the PAQI diagnosis: standard confidence-ranked initialization remains competitive without the additional selection strategy, supporting our decision to focus BUTR on budget allocation.

Table 7: DDQ-style initialization within D-FINE-L. Each variant is separately trained with its initializer, which is also used at inference. Original Top-K results are those reported in Tabs. 3 and 4.

Initialization	**AI-TOD-v2**		**VisDrone**	
	$\|Q\|$	AP	$\|Q\|$	AP
Original Top-K	600	35.2	400	29.6
DDQ-style	600	35.2	400	29.6

B Algorithmic Description of BUTR

Alg. 1 summarizes the BUTR query initialization module inserted after the encoder proposal generation stage of a two-stage DETR. Given the encoder proposals from the host detector, BUTR computes proposal-level ranking scores from classification logits and aggregates them into an image-level density proxy. This proxy is mapped to a decoder query budget through a safety factor; in practice, the budget can be optionally clipped to a feasible range imposed by the detector or comparison protocol. The resulting budget is then used in the standard Top-$K(x)$ initialization form. Therefore, BUTR modifies only the decoder query budget using existing encoder outputs and proposal confidences, without adding new learned modules or changing the backbone, encoder, decoder, or detection heads.

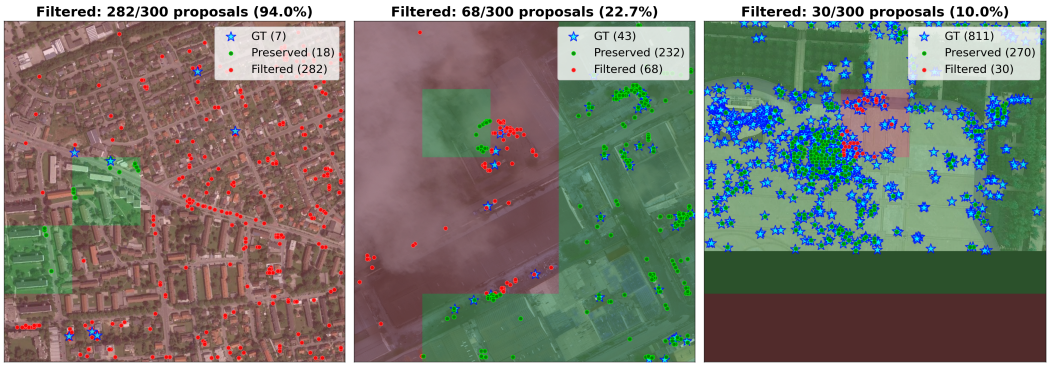


Figure 6: Qualitative examples of window-mask filtering. Green and red regions indicate mask-positive (preserved) and mask-negative (filtered) areas, respectively. Cyan stars denote GT centers; green/red points are preserved/filtered high-confidence proposals. Without Split, high-confidence proposals aligned with GT can be erroneously filtered.

Algorithm 1 BUTR Query Initialization

Require: Encoder proposals $P = \{(\boldsymbol{z}_i, \boldsymbol{r}_i, \boldsymbol{\ell}_i)\}_{i=1}^{L}$ from a two-stage DETR.
Require: Safety factor $\gamma \geq 1$; optional feasible range $1 \leq K_{\min} \leq K_{\max} \leq L$, with $L \geq 1$.
Ensure: Decoder query set Q.
// Proposal scoring
for $i = 1$ to L **do**
    Compute the proposal ranking score: $s_i \leftarrow \phi(\boldsymbol{\ell}_i) = \max_{c \in \{1,\dots,C\}} \sigma(\ell_{ic})$.
end for
// Density proxy estimation
Aggregate proposal scores as the image-level density proxy: $e(x) \leftarrow \sum_{i=1}^{L} s_i$.
// Budget mapping
Estimate the decoder query budget: $\widetilde{K}(x) \leftarrow \lceil \gamma e(x) \rceil$.
if a feasible range $[K_{\min}, K_{\max}]$ is specified **then**
    $K(x) \leftarrow \mathrm{clip}\big(\widetilde{K}(x), K_{\min}, K_{\max}\big)$.
else
    $K(x) \leftarrow \min\{L, \max\{1, \widetilde{K}(x)\}\}$.
end if
// Standard Top-K query initialization form
Select the indices of the top $K(x)$ proposals according to $\{s_i\}_{i=1}^{L}$: $\mathcal{I} \leftarrow \mathrm{TopK}(\{s_i\}_{i=1}^{L}, K(x))$.
Construct decoder queries from the selected proposals: $Q \leftarrow \{(\boldsymbol{z}_i, \boldsymbol{r}_i) \mid i \in \mathcal{I}\}$.
return Q.

C Qualitative Examples of Erroneous Filtering

Fig. 6 shows qualitative examples of the failure mode discussed in Sec. 3. Without SPLIT, the subsequent selection operators may filter out high-confidence proposals that are well aligned with GT objects. As shown in the figure, some proposals near GT centers fall into mask-negative regions and are erroneously discarded, which helps explain the large AP drops in Rows 4–6 of Tab. 1. These examples highlight the actual role of SPLIT: it protects top-ranked proposals from harmful filtering.

D Why Budget-Matched Top-K Reproduces PAQI

Two observations help explain the similar overall AP between PAQI and a Top-K initializer with an approximately matched average budget: overlap between the selected queries and redundancy in encoder proposals. The overlap analysis below matches the query count per image, whereas the AP comparison in Tab. 1 uses fixed budgets close to PAQI's average count.

Overlap. We first examine whether PAQI selects nearly the same queries as a confidence-ranked Top-$|Q|$ initializer. To this end, we define the overlap ratio as

$$r_{\text{overlap}} = \frac{|Q_{\text{PAQI}} \cap Q_{\text{Top-}|Q|}|}{|Q_{\text{PAQI}}|},$$

where Q_{PAQI} denotes the queries selected by PAQI and $Q_{\text{Top-}|Q|}$ denotes the confidence-ranked Top-$|Q|$ proposals from the same encoder outputs. The mean overlap is $71.5\% \pm 8.1\%$ on AI-TOD-v2 and $84.6\% \pm 1.9\%$ on VisDrone. For reference, the random baseline is $\mathbb{E}[|Q|/K_{\max}]$, with standard deviation induced by per-image variation in $|Q|$, yielding $42.0\% \pm 15.3\%$ on AI-TOD-v2 and $77.7\% \pm 4.2\%$ on VisDrone. These results show that PAQI is not identical to Top-$|Q|$, but already overlaps substantially with the confidence-ranked proposals, especially on VisDrone.

Redundancy. High overlap alone does not fully explain the similarity in overall AP. Another contributing factor is the intrinsic redundancy of encoder proposals. As shown in Fig. 7, even under different budgets, GT objects are typically already surrounded by nearby proposals, while increasing K mainly makes these proposals denser rather than introducing entirely new object coverage. This suggests that encoder proposals often provide multiple candidate locations for the same object, especially around foreground regions. Therefore, once the decoder receives a sufficiently large set of high-confidence proposals, increasing the budget further mainly adds redundant evidence rather than qualitatively new information.

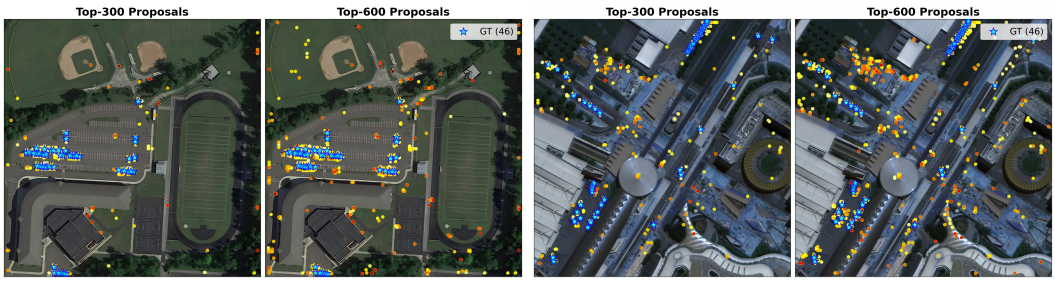


Figure 7: Qualitative examples of encoder proposal redundancy using different budgets. Cyan stars denote GT centers, and proposal points are colored from yellow to red by confidence. Even with Top-300, most GT objects are already covered by nearby proposals; increasing the budget to Top-600 mainly increases proposal density and redundancy rather than introducing new object coverage. This redundancy helps explain why budget-matched Top-K can closely match PAQI in overall AP.

Taken together, these observations help explain why a plain Top-K initializer with sufficient capacity can closely reproduce PAQI's overall AP. They support the budgeting interpretation without establishing that arbitrary query identities are interchangeable or that selection is unnecessary in all settings.

E Full Density-Stratified Results

To complement Sec. 5.3, we provide the full density-stratified results in Fig. 8 and Fig. 9, including the tail subsets already reported in the main text. BUTR's gains are observed in multiple lower-density subsets as well as the tail, although it does not outperform every baseline in every model-scale–subset comparison. These results describe performance across the density distribution rather than a decomposition of the overall AP gain.

F Stability of Tail Performance Across Training Runs

To assess the stability of performance on the small high-density tail, we conduct two complementary evaluations of Dome-DETR-L and BUTR-L on AI-TOD-v2 under matched training and evaluation settings. We report AP^{ext} on the same `extreme` subset across runs.

Evaluation protocols. The main results use the predefined seed 0. For fixed-seed reproducibility, we perform three independent training runs with seed 0, including the originally reported run. For multi-seed robustness, we compare the two methods using matched seeds 0, 1, and 2; the seed-0 entry is the originally reported run and is shared with the fixed-seed evaluation. For each run, we select the checkpoint with the highest overall AP and evaluate it separately on the `extreme` subset to obtain AP^{ext}. Both evaluations report the mean and standard deviation over three runs.

Table 8: Fixed-seed reproducibility of AP^{ext} on AI-TOD-v2: three independent seed-0 training runs. Run 1 is the originally reported result. Mean and standard deviation describe variation across training runs.

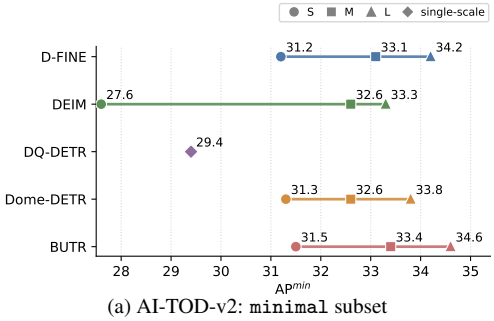
Method	Run 1	Run 2	Run 3	AP^{ext}
Dome-DETR-L	29.9	29.1	28.8	29.27 ± 0.57
BUTR-L	29.8	29.7	29.4	$\mathbf{29.63 \pm 0.21}$

Fixed-seed reproducibility. As shown in Tab. 8, BUTR-L exhibits lower AP^{ext} variation under the original seed-0 protocol: 29.63 ± 0.21, compared with 29.27 ± 0.57 for Dome-DETR-L.

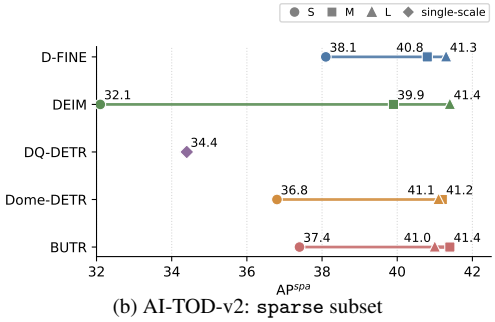
Table 9: Multi-seed robustness of AP^{ext} on AI-TOD-v2 using matched seeds 0, 1, and 2. Seed 0 is the originally reported result. Mean and sample standard deviation describe variation across seeds.

Method	Seed 0	Seed 1	Seed 2	AP^{ext}
Dome-DETR-L	29.9	27.2	27.5	28.20 ± 1.48
BUTR-L	29.8	29.3	29.3	$\mathbf{29.47 \pm 0.29}$

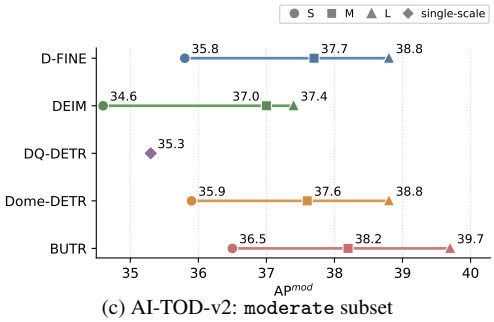
Multi-seed robustness. Tab. 9 shows that BUTR-L achieves higher mean AP^{ext} with lower standard deviation (29.47 ± 0.29 versus 28.20 ± 1.48).



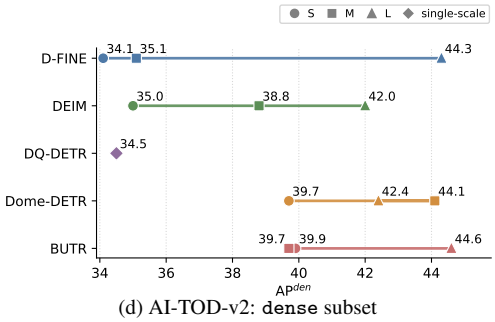
(a) AI-TOD-v2: `minimal` subset



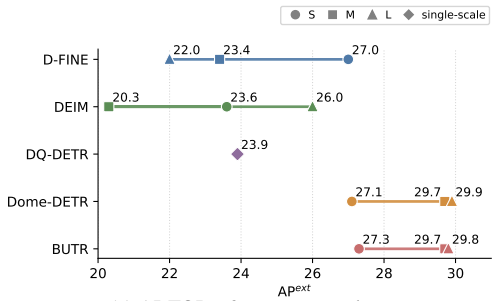
(b) AI-TOD-v2: `sparse` subset



(c) AI-TOD-v2: `moderate` subset

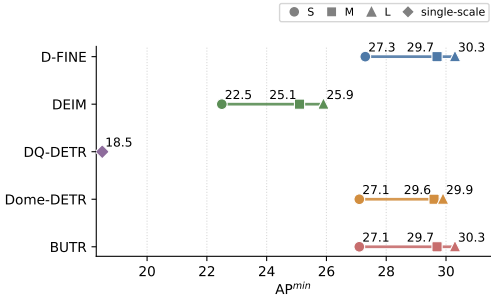


(d) AI-TOD-v2: `dense` subset

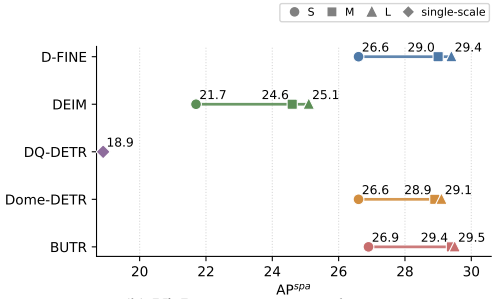


(e) AI-TOD-v2: `extreme` subset

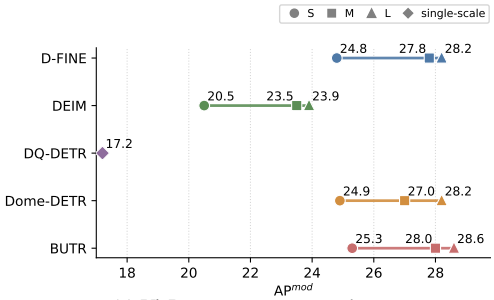
Figure 8: Full density-stratified results on AI-TOD-v2, including the main-text `extreme` subset.



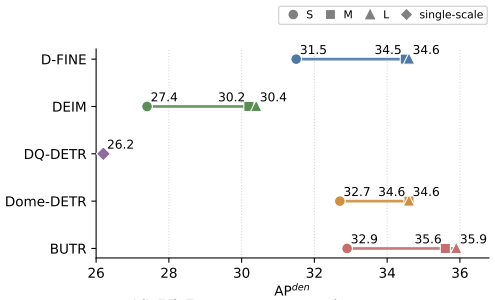
(a) VisDrone: `minimal` subset



(b) VisDrone: `sparse` subset



(c) VisDrone: `moderate` subset



(d) VisDrone: `dense` subset

Figure 9: Full density-stratified results on VisDrone, including the main-text `dense` subset.

Together, these results show lower observed variability in BUTR-L's tail performance under both fixed-seed repetitions and different random seeds in this configuration. In light of the erroneous-filtering examples in App. C, this stability advantage may partly stem from avoiding errors introduced by Dome-DETR's learned spatial mask: BUTR instead uses a non-learned budgeter to adjust the query count while retaining standard confidence-ranked initialization.

G Supplementary Experiment on Dense Agricultural Imagery

As a supplementary cross-domain evaluation, we use Sorghum-Grain-Count [31], which contains many small, visually similar grain instances outside the aerial domain.

Data preparation. The original training and validation splits are combined into a training set, while the original test split is reserved for evaluation. We crop the 3456×5184 source images into overlapping 800×800 patches with a stride of 640 and retain patches containing at least one annotated grain. Tab. 10 summarizes the resulting cropped splits. Annotation totals and objects-per-patch statistics refer to the retained patches, rather than counts of unique grains across the original images. This protocol evaluates detection on nonempty crops; it does not establish performance on empty-background images or full-image counting after merging overlapping predictions.

Table 10: Statistics of the cropped Sorghum-Grain-Count splits. Objects/patch reports the mean and standard deviation across retained patches. P90, P95, and P99 denote percentiles of the patch-level object-count distribution.

Split	Patches	Annotations	Objects/patch	Median	Min–Max	P90	P95	P99
Trainval	1,859	178,238	95.88 ± 89.32	70.0	1–429	229.0	272.0	350.4
Test	433	45,341	104.71 ± 94.31	75.0	1–523	236.8	277.4	361.3

Comparison protocol and results. Within this comparison, all models use matched training and evaluation settings with 800×800 inputs. We evaluate D-FINE-L and DEIM-L with fixed budgets of 400 and 1100 queries, close to the observed average query counts of BUTR-L (378) and Dome-DETR-L (1103), respectively. As shown in Tab. 11, raising the fixed budget from 400 to 1100 improves AP from 57.0 to 57.7 for D-FINE-L and from 57.2 to 58.0 for DEIM-L. BUTR-L achieves 58.0 AP with an average of only 378 queries, exceeding D-FINE-L and matching DEIM-L at the larger budget while requiring less training time. It also improves over Dome-DETR-L by 0.4 AP while using approximately one-third as many queries, with $2.2\times$ faster training. These results provide supplementary evidence for confidence-based budgeting in a dense agricultural setting.

Table 11: Comparison on Sorghum-Grain-Count using the cropped trainval/test protocol. Query-count mean and standard deviation describe variation across evaluation patches, not repeated training runs. Tr. (h): training time. T4 deployment latency was not measured for this supplementary experiment.

Method	Tr. (h)	Init.	$\lvert Q\rvert$ $(\mu \pm \sigma)$	AP	AP_{50}	AP_{75}	AP_t	AP_s	AP_m
D-FINE-L	1.5	Fixed	400	57.0	92.6	64.9	10.6	50.1	63.6
D-FINE-L	1.7	Fixed	1100	57.7	93.9	65.6	13.4	50.9	64.2
DEIM-L	1.6	Fixed	400	57.2	92.9	65.1	10.9	50.3	63.8
DEIM-L	1.8	Fixed	1100	**58.0**	94.0	66.3	14.3	51.3	64.3
Dome-DETR-L	3.3	Dynamic	1103 ± 195	57.6	94.0	65.4	13.1	50.8	64.1
BUTR-L	1.5	Dynamic	378 ± 119	**58.0**	93.8	66.1	14.5	51.5	64.1

H T4 Deployment Latency Protocol

Measurement protocol. We use a single NVIDIA Tesla T4 (16 GB) on Kaggle, Python 3.10.14, PyTorch 2.4.0+cu118, and TensorRT 10.4.0, with batch size 1. Inputs are 800×800 for tiny-object benchmarks and 640×640 for COCO. Paired comparisons run sequentially on the same physical GPU. Each result averages three rounds, each comprising 1,000 warmup calls and the same ordered 1,000-image subset of the corresponding test split (val2017 for COCO); all samples and rounds are retained. We use the D-FINE/DEIM synchronized timing boundary with synchronous `execute_v2`, without CUDA Graphs. The timer includes all model stages, detection post-processing, and dynamic-budget computation, readback, synchronization, and shape handling. Image loading, preprocessing, and input transfer are excluded.

Precision and layout. Tiny-object engines enable FP16 with sensitive operations constrained to FP32; DQ-DETR additionally protects bounding-box and sampling-coordinate operations. TF32 is disabled. COCO uses the official default `trtexec --fp16` build settings. BUTR uses separate encoder/decoder engines and splits cross-attention value features by spatial scale before transposing each scale. This layout rewrite preserves the budgeting rule and decoder mathematics. D-FINE/DEIM retain their original single-engine graphs; Dome-DETR and DQ-DETR use front/back engines, with all stages included in timing.

Layout fairness controls. Applying the same rewrite to complete AI-TOD-v2 D-FINE-S/DEIM-S graphs increases latency from 12.14/12.23 ms for independently rebuilt original graphs to 15.17/15.32 ms. We therefore retain the faster original baseline layouts. These matched controls show that the rewrite's benefit depends on the surrounding graph and engine boundaries, rather than establishing a universal speedup.

COCO calibration. For official D-FINE-S/M/L/X checkpoints with 300 queries, we measure 3.624/5.635/7.909/13.485 ms, versus published values of 3.49/5.62/8.07/12.89 ms; all deviations are within 5%. The physical T4 is selected using the predeclared mean absolute relative deviation across all four calibration means (2.68%), without using BUTR performance. Paired COCO plug-in measurements use this same card.

Accuracy scope. AP is reported from the detection experiments, rather than a full-test FP16 deployment evaluation. BUTR's paired layout checks preserve per-image budgets and change subset AP by less than 0.04 points across both tiny-object datasets. AI-TOD-v2 DEIM-S timing remains provisional because of an unresolved deployment-accuracy mismatch. VisDrone DEIM-S/M uses same-scale D-FINE weights with the DEIM inference architecture. Full deployment scripts and additional checks are provided in the repository.

I Broader Impact

The broader impact of this work is primarily methodological. By reframing dynamic query as a budgeting problem, we aim to clarify the mechanism behind adaptive query allocation and encourage simpler, more principled designs in tiny-object detection. Our results also suggest that density-stratified evaluation can serve as a useful complementary protocol when overall AP obscures behavior on rare but important high-density cases. Beyond tiny-object detection, the COCO plug-in results suggest that simple query budgeting can also be useful in general-purpose detection.

At the application level, improving efficiency and reliability in dense and tiny-object scenarios may benefit tasks such as remote sensing, traffic analysis, and environmental monitoring. At the same time, stronger detection in aerial or crowded scenes could also be misused for intrusive surveillance or monitoring, so downstream deployment should still consider appropriate privacy, legal, and ethical safeguards.

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 state the paper's main contributions: a mechanism-oriented diagnosis of dynamic query, the budgeting view, the proposed BUTR framework, and evaluations on AI-TOD-v2, VisDrone, and COCO. The claims are supported by the diagnostic analyses, ablations, and experimental results reported in the paper.

   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: The paper discusses the scope and limitations of the approach, including its focus on two-stage DETR-style detectors, the use of encoder confidences as a density proxy, and the empirical nature of the evaluation across the considered datasets and baselines.

   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: The paper does not present theoretical results, theorems, or formal proofs. The proposed budgeting principle is empirically motivated and evaluated through experiments.

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: The paper specifies the datasets, splits, evaluation protocol, model variants, training schedules, query budgets, latency measurement protocol, hardware, and implementation details needed to reproduce the main results. The BUTR procedure is also described algorithmically in the appendix.

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: The experiments are conducted on publicly available datasets and based on publicly available detector implementations/checkpoints. We provide code and instructions for reproducing the main experiments in the supplementary material.

   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: The paper reports the datasets and splits, input resolution, training schedules, query budgets, evaluation settings, evaluation protocol, latency measurement protocol, and hardware configuration.

   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: The main detection results are reported as standard single-run COCO-style AP evaluations, following common practice. For diagnostic analyses where applicable, the paper reports mean and standard deviation over images, such as query counts, overlap ratios, and query-usage behavior.

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 paper reports the compute platform, including CPUs, GPUs, input resolution, batch size, training-time measurement protocol, and per-image inference latency. Training wall-clock time and inference latency are reported for the main comparisons.

   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: The research uses public datasets and standard object detection benchmarks, and does not involve deceptive data collection, human-subject experiments, or privacy-invasive data collection by the authors. We have reviewed the NeurIPS Code of Ethics and believe the work conforms to it.

   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: [Yes]

    Justification: The paper discusses potential positive impacts, such as improving the efficiency and reliability of tiny-object detection, as well as potential negative impacts, including misuse of improved aerial or dense-scene detection systems for surveillance or monitoring. The method is a general detection technique and is not deployed in any real-world system in this work.

    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 does not release high-risk generative models, scraped datasets, or models specifically designed for misuse. The work uses existing public detection datasets and focuses on algorithmic analysis and efficiency.

    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: The paper cites the datasets, baseline methods, and detector implementations/checkpoints used in the experiments. Existing assets are used according to their respective licenses and terms of use.

    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: The paper introduces code for the proposed BUTR budgeter and evaluation scripts. The released assets are documented with instructions for reproducing the main experiments and are anonymized for submission when applicable.

    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: The paper does not involve crowdsourcing, user studies, 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: The paper does not involve crowdsourcing or research with human subjects, so IRB approval or equivalent review is not applicable.

    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: The core method development, experiments, and scientific claims do not involve LLMs as an important, original, or non-standard component. Any LLM use, if any, was limited to writing, editing, or formatting assistance and did not affect the methodology, scientific rigor, or originality of the research.

    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.