WINARDET: COMPRESSED WINDOW AUTO-REGRESSION FOR LONG-HORIZON VISION-LANGUAGE GROUNDING

Anonymous authors
Paper under double-blind review

ABSTRACT

MLLM-based detectors formulate vision-language grounding as autoregressive coordinate generation, yet in dense scenes, they often collapse into a box repetition loop, repeatedly emitting identical boxes until the decoding budget is exhausted. We associate this failure with a historical-answer attention takeover: Under full-history autoregression, repeated coordinate tokens in the growing answer history attract disproportionate attention, crowding out the attention sinks needed to attend to new visual evidence. To break this loop, we propose **Compressed Window Auto-Regression** (WinAR). This new mechanism partitions a long answer sequence into consecutive generation windows and restricts full token-level autoregression to the current window only while compressing each previous window into a fixed set of latent tokens that condition subsequent decoding. This design reduces the repeated exposure to raw historical coordinates that triggers the attention takeover, yet preserves cross-window dependencies essential for coherent multi-object generation. On RSDense, which contains 200-2,000 objects per image, our model WinARDet improves RexOmni F1-score (IoU=0.5) from 32.16 to 39.36 over the baseline, while reducing the redundancy rate from 35.48% to 13.00%. WinARDet also generalizes across decoding strategies and output formats, including parallel box decoding and polygon generation, providing a practical paradigm for long-horizon vision-language grounding across scene densities.

1 INTRODUCTION

Modern MLLM-based detectors, such as RexOmni (Jiang et al., 2026) and LocateAnything (Wang et al., 2026b), formulate object detection as autoregressive next-token prediction, enabling flexible vision-language grounding under complex instructions beyond the fixed prediction interfaces of conventional detectors (Liu et al., 2024; Cheng et al., 2024; Wang et al., 2026a). However, extending this autoregressive formulation to dense visual scenes introduces a fundamental long-horizon grounding challenge (Xia et al., 2018; Goldman et al., 2019). When hundreds or even thousands of objects must be generated within a single response, the model must continuously discover previously uncovered objects while keeping track of those already localized. Since autoregressive inference operates under a finite and costly decoding budget, every redundant prediction consumes generation steps that could otherwise be used to recover new objects. Sustaining new-object discovery throughout a long decoding horizon is therefore critical to both detection accuracy and inference efficiency.

Our empirical investigation reveals that MLLM-based detectors under the Full-History Auto-Regression (FHAR) paradigm, where each prediction conditions on the complete history of generated answer tokens, progressively lose their ability to sustain new-object discovery over extended decoding horizons. In dense configurations, we observe that a supervised fine-tuned RexOmni model (Jiang et al., 2026) under the FHAR paradigm initially yields valid detections, yet eventually deteriorates into a pathological box repetition loop, wherein identical or nearly identical bounding boxes are emitted repeatedly (Fig. 1(middle, top)). This decoding collapse is quantitatively characterized in Fig. 2(a): Over the initial 400 box predictions, the redundancy rate escalates from approximately 10% to near 100%, while the rate of discovering novel objects concomitantly plummets from roughly 16% to nearly zero. Consequently, a substantial portion of the remaining decoding budget is expended on redundant box predictions rather than on unaccounted objects, causing recall

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

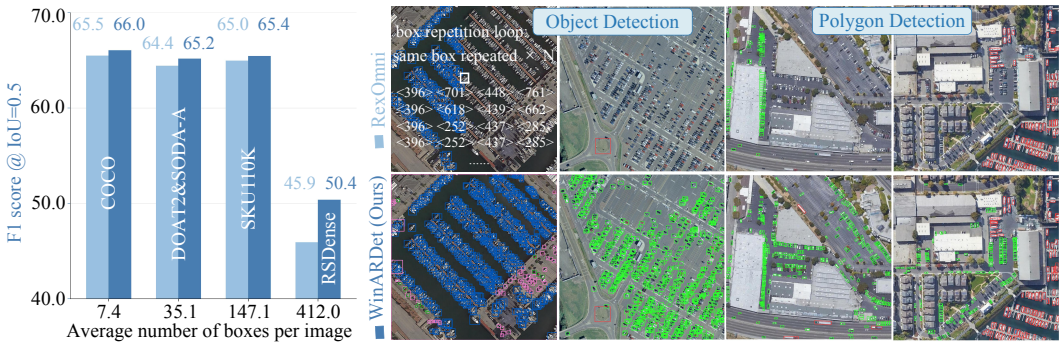


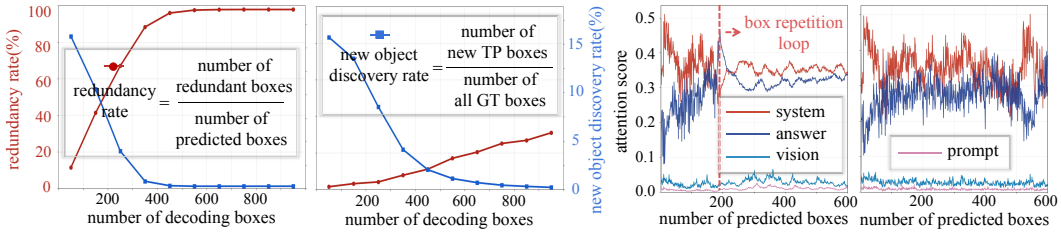
Figure 1: WinARDet overview. **Grounding Performance across Densities(Left):** WinARDet outperforms the baseline across benchmarks of varying scene densities(Lin et al. (2014); Xia et al. (2018); Cheng et al. (2023); Goldman et al. (2019)), with increasingly pronounced gains in denser scenes. **Breaking Box Repetition Loop(Middle):** WinARDet breaks the box repetition loop, suppressing repeated predictions and recovering substantially more distinct objects. **Beyond Bounding Boxes(Right):** WinARDet generalizes to polygon detection, extending beyond bounding-box.

to saturate while inference latency continues to accumulate. Ultimately, this degenerative process substantially undermines both the detection accuracy and the overall inference efficiency.

Our attention analysis links this degeneration to a progressive historical-answer attention takeover. Specifically, we partition the conditioning context into system, visual, prompt, and historical-answer regions, and track the softmax-normalized attention mass assigned to each region by coordinate-token queries throughout decoding. As shown in Fig. 2(b), attention to the historical answer grows steadily with the decoding horizon and rises sharply near the onset of the repetition loop, accompanied by a pronounced reduction in attention to the initial system tokens. This redistribution is particularly revealing because prior studies (Xiao et al., 2024; Qiu et al., 2026) have shown that a small set of initial system tokens often serves as attention sinks, absorbing excess attention during autoregressive decoding. In dense grounding, however, the expanding raw answer history progressively displaces this sink allocation with explicit and increasingly repetitive coordinate patterns. Once these raw coordinates dominate the conditioning context, the decoder becomes increasingly biased toward previously generated locations. Each duplicated box is then appended back to the history, further strengthening this bias and ultimately forming a self-reinforcing box repetition loop.

To break this loop, we propose **Compressed Window Auto-Regression** (**WinAR**), an alternative autoregressive paradigm for long-horizon structured generation. WinAR follows a simple principle: *remember compactly and generate locally*. In contrast to FHAR, which retains every previous coordinate token, WinAR preserves exact token-level history only within the current window and represents each previous completed window with a fixed number of latent tokens by a Window History Compressor. During training, preceding windows appear only through their compact latent tokens, and cross-entropy supervision is applied to the current window. During inference, each completed window is compressed before generation proceeds to the next. WinAR therefore changes how autoregressive context is carried across a long answer: local generation remains exact, while cross-window dependencies pass through a compact history. By limiting repeated exposure to raw coordinate patterns without eliminating historical conditioning, WinAR mitigates historical-answer attention takeover and preserves the decoding budget for discovering new objects.

Experiments show that the benefit of WinAR becomes increasingly pronounced as scene density grows (Xia et al., 2018; Cheng et al., 2023). On RSDense benchmark, which contains 200–2,000 objects per image, integrating WinAR into RexOmni reduces the mean prediction redundancy rate from 35.48% to 13.00% and improves F1@IoU$=$ 0.5 from 32.16 to 39.36. It also generalizes to the parallel box decoding framework of LocateAnything (Wang et al., 2026b) and improves F1 from 26.15 to 27.54 and reduces mean redundancy from 14.89% to 2.13% on RSDense. WinAR also extends to structured outputs such as polygons and substantially reduces peak training memory. When training LocateAnything, WinAR reduces peak GPU memory consumption by 34.1%, enabling training on 8 NVIDIA A40 GPUs(48 GB) without MagiAttention (Zewei & Yunpeng, 2025). These results establish WinAR as a general and practical paradigm for accurate and efficient vision-language grounding across varying scene densities. Our contributions are threefold:



(a) Redundancy and New-Object Discovery over Decoding (b) Attention Dynamics over Decoding

Figure 2: **(a)** Metrics are averaged over intervals of 100 predicted boxes on RSDense(Section 4). FHAR(left) rapidly accumulates redundant predictions as new-object discovery collapses, whereas WinAR(right) suppresses redundancy and sustains the discovery of new objects. **(b)** The onset of the box repetition loop coincides with historical-answer attention takeover and reduced system-token attention in FHAR(left), while WinAR(right) maintains a more stable attention allocation.

- We identify a long-horizon failure mode in MLLM-based detectors, where Full-History Auto-Regression progressively triggers historical-answer attention takeover and a self-reinforcing box repetition loop.
- We propose **Compressed Window Auto-Regression (WinAR)**, which preserves exact autoregression within local windows while compressing completed windows into compact latent tokens, mitigating repetition without discarding cross-window dependencies.
- Extensive experiments across scene densities, decoding strategies, and structured output formats demonstrate that WinAR consistently improves detection accuracy, substantially reduces prediction redundancy, and lowers training memory for long-sequence grounding.

2 Related Work

Regression-based Object Detection (Ren et al., 2016; Cai & Vasconcelos, 2018; Carion et al., 2020; Zhu et al., 2020; Redmon et al., 2016; Dai et al., 2021) is built upon a discriminative classification and regression paradigm. They typically extract multi-scale feature maps with a backbone (Dosovitskiy, 2020; Siméoni et al., 2025) and a feature pyramid network (Lin et al., 2017a), instantiate candidate proposals in the form of anchors (Ren et al., 2016), points (Tian et al., 2019), or learnable object queries (Carion et al., 2020), assign foreground and background samples via label assignment strategy (Zhang et al., 2020; Wang et al., 2025a), optimize semantic recognition and geometric localization with classification (Lin et al., 2017b) and regression losses (Li et al., 2020), respectively. Motivated by large-scale contrastive vision-language pre-training such as CLIP (Radford et al., 2021), open-vocabulary detectors (Li et al., 2022; Liu et al., 2024) further align category names or textual prompts with regional visual features, extending detection beyond fixed class IDs. However, their language understanding remains limited by relatively lightweight text encoders (Devlin et al., 2019) and a narrow supervision interface dominated by region–text matching with category names or short noun phrases. As a result, they struggle to exploit heterogeneous vision-language supervision and remain limited in complex language instruction understanding and reasoning tasks.

MLLM-based Object Detection (Chen et al., 2023; Wang et al., 2025b; Guo et al., 2025; Bai et al., 2025; Li et al., 2025; 2026; Jiang et al., 2026; Wang et al., 2026b) leverages pretrained large language models to unify heterogeneous vision-language tasks under a shared generative objective, enabling detection conditioned on complex semantics and language instructions. As an early precursor, Pix2Seq (Chen et al., 2021) quantizes object categories and box coordinates into discrete tokens and casts detection as auto-regressive sequence prediction. This formulation is also extended to MLLMs, allowing object detection, pointing, OCR, keypoint localization, and other vision-language tasks to be expressed through a shared generative interface. RexOmni (Jiang et al., 2026) represents geometric locations using special coordinate tokens and adopts large-scale supervised fine-tuning followed by geometry-aware GRPO post-training to improve localization accuracy and mitigate coordinate drift and duplicate predictions. LocateAnything (Wang et al., 2026b) further introduces parallel box decoding, which treats each bounding box or point as an atomic geometric unit, preserving intra-box coherence while reducing the serial latency of token-by-token coordinate generation. Despite these

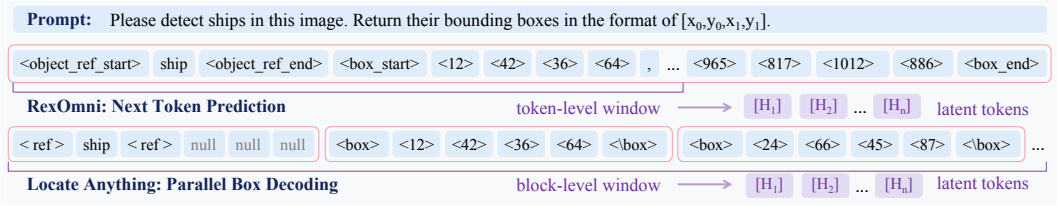


Figure 3: Illustration of the serialized response formats and compression granularities for RexOmni and LocateAnything. RexOmni represents the answer as a flat token sequence and generates it through next-token prediction, whereas LocateAnything organizes the answer into box-aligned blocks and predicts the coordinate tokens within each block in parallel. Accordingly, WinAR partitions RexOmni outputs into fixed-length token windows (e.g., every 300 tokens), while forming windows from a fixed number of complete blocks (e.g., every 50 blocks) for LocateAnything.

advances, these works still preserve the ever-growing raw answer history during auto-regressive generation and do not explicitly address long-horizon dense grounding, where accumulated coordinate tokens can dominate subsequent predictions and induce self-reinforcing box repetition loop.

3 METHOD

We propose **Compressed Window Auto-Regression (WinAR)**, a simple and efficient paradigm for long-horizon vision-language grounding that replaces Full-History Autoregression with windowed generation conditioned on compact historical context. Section 3.1 introduces the preliminaries of autoregressive vision-language grounding based on RexOmni (Jiang et al., 2026) and LocateAnything (Wang et al., 2026b), followed by a detailed description of WinAR in Section 3.2.

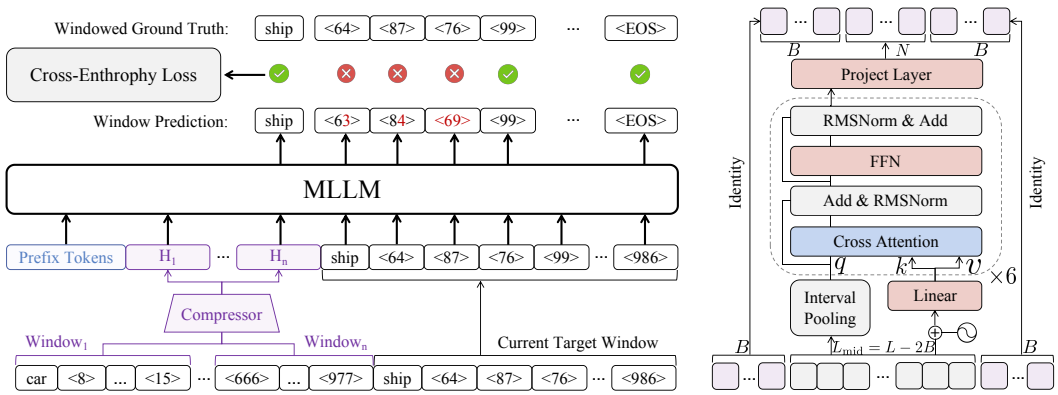
3.1 PRELIMINARIES

MLLM-based detectors are built upon pretrained multimodal large language models, such as Qwen2.5-VL-3B. Given an image I and a natural-language prompt P, the model encodes I into visual tokens $\mathbf{V}$ using a visual encoder (Dosovitskiy, 2020), while the system instruction and prompt are tokenized and mapped into text embeddings. The resulting system, visual, and prompt tokens are concatenated to form the multimodal prefix $\mathbf{X} = [\mathbf{S}; \mathbf{V}; \mathbf{P}]$. Under Full-History Auto-Regression, the target answer $\mathbf{Y} = (y_1, \ldots, y_T)$ is generated through next-token prediction conditioned on the multimodal prefix $\mathbf{X}$ and the complete raw answer history:

$$p_\theta(\mathbf{Y} \mid \mathbf{X}) = \prod_{t=1}^{T} p_\theta\left(y_t \mid \mathbf{X}, \mathbf{Y}_{<t}\right), \tag{1}$$

where $\mathbf{Y}_{<t}$ denotes all previously generated answer tokens. During supervised fine-tuning, the ground-truth answer is appended to the multimodal prefix $\mathbf{X}$ and optimized with cross-entropy loss under a causal attention mask. At inference time, the model first prefills the multimodal prefix $\mathbf{X}$ into the KV cache, and then autoregressively generates answer tokens while incrementally updating the cache until an end-of-sequence token or the maximum decoding length is reached.

To adapt the general MLLM generation paradigm to object detection, RexOmni (Jiang et al., 2026) and LocateAnything (Wang et al., 2026b) recast object detection as structured sequence prediction over category and coordinate tokens. Both methods encode spatial locations using quantized relative coordinates. Horizontal and vertical pixel coordinates are normalized by the image width and height, respectively, and uniformly quantized into 1,000 discrete bins. The final 1,000 entries of the language-model vocabulary, denoted by $\langle 0 \rangle$ through $\langle 999 \rangle$, are repurposed as coordinate tokens, allowing a bounding box $[x_0, y_0, x_1, y_1]$ to be represented by four coordinate tokens. As illustrated in Fig. 3, RexOmni further uses the original structural tokens of Qwen2.5-VL-3B, such as `<|object_ref_start|>` and `<|box_end|>`, to delimit each category phrase and its associated boxes, and serializes the complete answer into a flat sequence for standard next-token prediction. LocateAnything instead reorganizes the response into fixed-length atomic blocks of six tokens to support parallel box decoding. Different block types are used to represent category phrases and box



(a) Supervised fine-tuning with WinAR (b) Window History Compressor

Figure 4: Overview of the proposed Compressed Window Auto-Regression. **(a)** A long answer is partitioned into consecutive windows and processed sequentially. Preceding windows are compressed into memory and combined with the multimodal prefix to condition the current window, on which cross-entropy loss is applied. **(b)** Structure of the Window History Compressor.

outputs, while unused positions are padded with `<null>` tokens to maintain a uniform block length. Under this formulation, each box can be decoded in parallel within a single forward pass, reducing decoding latency while ensuring box-aligned and structural outputs.

3.2 Compressed Window Auto-Regression

Motivated by the historical-answer attention takeover and box repetition loop analyzed in Section 1, WinAR replaces the ever-growing raw answer history of FHAR with windowed generation conditioned on compact historical context. As illustrated in Fig. 4(a), WinAR follows a simple principle: *remember compactly and generate locally*. Given a target answer $\mathbf{Y} = (y_1, \ldots, y_T)$ of length T and a predefined window length L, we partition $\mathbf{Y}$ into $K = \lceil T/L \rceil$ consecutive windows, where each window contains L tokens except the last:

$$\mathbf{Y} = [\mathbf{Y}^{(1)}; \mathbf{Y}^{(2)}; \ldots; \mathbf{Y}^{(K)}], \tag{2}$$

where $\mathbf{Y}^{(k)}$ denotes the k-th window. Before generating the k-th window, each completed window is independently transformed by the Window History Compressor into latent tokens:

$$\mathbf{H}^{(i)} = \mathcal{C}_\phi\left(\mathbf{Y}^{(i)}\right), \qquad i < k, \tag{3}$$

where $\mathcal{C}_\phi$ denotes the Window History Compressor, which is detailed in the following paragraph. The resulting generation process is then formulated as:

$$p_\theta(\mathbf{Y}_t^{(k)} \mid \mathbf{X}) = \prod_{t=1}^{T_k} p_\theta\left(y_t^{(k)} \mid \mathbf{X}, \mathbf{H}_{<k}, \mathbf{Y}_{<t}^{(k)}\right), \tag{4}$$

where only the local history $\mathbf{Y}_{<t}^{(k)}$ of the current window remains explicitly visible. The original multimodal prefix $\mathbf{X}$ is retained throughout generation, while the answer context grows only through compact latent tokens rather than raw coordinate token sequences.

Window History Compressor transforms each completed answer window of length L into a compact set of latent tokens. Since fixed-length windowing may split a category phrase or a geometric structure at the window boundary, we preserve B tokens at both ends of the window to retain their exact local context. Thus the k-th window is decomposed as $\mathbf{Y}^{(k)} = \left[\mathbf{y}_{\text{front}}^{(k)}; \mathbf{y}_{\text{mid}}^{(k)}; \mathbf{y}_{\text{back}}^{(k)}\right]$, where the front and back portions each contain B boundary tokens and are passed directly to the output without compression, while the middle portion contains the remaining $L_{\text{mid}} = L - 2B$ tokens and is the only part processed by the compressor. As illustrated in Fig. 4(b), given a predefined number of compressed latent tokens N, the middle tokens are processed through two complementary branches.

In the first branch, the $(L - 2B)$ middle tokens are uniformly partitioned into N contiguous intervals, and each interval is average-pooled to initialize one query. This content-aware initialization provides each query with a localized summary of a distinct portion of the window, rather than relying on randomly initialized queries. To enable equal partitioning, the hyperparameters L, B, and N are chosen such that $L - 2B$ is divisible by N. In the second branch, learnable window-local positional embeddings are added to preserve relative order within the window, followed by a linear projection that produces the keys and values for cross-attention.

The queries are then refined through stacked cross-attention and feed-forward layers. The resulting tokens are normalized and passed through an output projection consisting of two linear layers with a GELU activation in between. Finally, the compressed latent tokens are concatenated with the front tokens $\mathbf{y}^{(k)}_{\text{front}}$ and back tokens $\mathbf{y}^{(k)}_{\text{back}}$. The compressor thus reduces an L-token window to $2B + N$ latent tokens, preserving exact boundary context while compactly summarizing the middle portion.

Training and Inference. During training, the ground-truth answer is processed window by window. For the k-th training stage, all preceding windows are represented solely by $\mathbf{H}_{<k}$, while cross-entropy supervision is applied only to the current target window. Raw windows can be discarded after compression, avoiding the need to retain full-sequence activations throughout training and thereby substantially reducing GPU memory consumption. As illustrated in Fig. 3, windows are formed using a fixed number of answer tokens for RexOmni, whereas for LocateAnything, window boundaries are aligned with complete atomic blocks.

At inference time, the model first generates $\mathbf{Y}^{(1)}$ from the multimodal prefix $\mathbf{X}$. Once the current window is completed, it is compressed into $\mathbf{H}^{(1)}$ and removed from the explicit answer context. The model then generates the next window conditioned on the original prefix, the accumulated memory tokens, and the locally generated tokens within the current window. This process repeats until an end-of-sequence token or the maximum decoding budget is reached.

4 Experiment

4.1 Training Details and Evaluation Setup

Training Details. For RexOmni, all variants are trained for 4 epochs with a per-device batch size of 1 and 16 gradient-accumulation steps. The base learning rate is 4×10^{-5}, with a vision-tower learning rate of 4×10^{-6}. **RexOmni-WinAR** additionally optimizes the Window History Compressor with a learning rate of 4×10^{-4} and adopts token-level windowing with $L = 300$, preserving $B = 6$ tokens at each boundary and compressing the remaining tokens into $N = 8$ latent tokens. For polygon detection, we use a larger window configuration with $L = 400$, $B = 8$, and $N = 12$. To control for the increased exposure of long samples introduced by window-wise training, we further construct a **RexOmni-FHAR*** baseline, where a sample containing T answer tokens is oversampled approximately $\lceil T/L \rceil$ times, matching both the effective sample exposure and the total number of optimizer updates of RexOmni-WinAR. We also conduct GRPO experiments following the reward design of RexOmni, training for one epoch on the subset of training samples containing more than 200 objects. For LocateAnything, we adopt the same optimization settings as RexOmni and construct corresponding **FHAR*** and **WinAR** variants. Since LocateAnything operates on box-aligned atomic blocks, LocateAnything-WinAR adopts block-level windowing with $L = 50$ blocks (300 tokens), preserves $B = 1$ block (6 tokens) at each boundary, and compresses the remaining blocks into $N = 2$ latent blocks (12 tokens). All FHAR-based experiments are conducted on eight NVIDIA H100 GPUs, whereas all WinAR experiments are conducted on eight NVIDIA A40 GPUs. More implementation details are provided in Section A.2.

Evaluation Setup. We conduct experiments on DOTA2 (Xia et al., 2018) and SODA-A (Cheng et al., 2023), jointly training on the union of their training sets and evaluating on the union of their validation sets, denoted as **DOTA2 & SODA-A**. Following RexOmni (Jiang et al., 2026), we report category-averaged Recall, Precision, and F_1 score at an IoU threshold of 0.5. For polygon detection, we train separate models on the DOTA2 and SODA-A training sets and evaluate them on their corresponding validation sets. To further evaluate performance under extremely dense conditions, we select all evaluation images in DOTA2 & SODA-A containing more than 200 objects to construct the **RSDense** subset and report image-averaged Recall, Precision, and F1 score at an IoU threshold of 0.5. To quantify prediction redundancy, we further report the duplicate rate $R\text{dup} = N_{\text{dup}}/N_{\text{pred}}$

Table 1: Comparison on DOTA2 & SODA-A and the extremely dense RSDense subset. R, P, and F1 denote recall, precision, and F1 score at an IoU threshold of 0.5. Mean R_{dup} and Mean R_{over} measure duplicate predictions and over-prediction, respectively. The best and second-best results within each comparison group are highlighted in **bold** and underline, respectively. NTP denotes next token prediction, PBD denotes parallel box decoding, and Hybrid denotes a mixed decoding mode combining NTP and PBD. See Locate Anything (Wang et al., 2026b) for details.

Method	Mode	DOTA2 & SODA-A					RSDense				
		R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5	Mean R_{dup} ↓	Mean R_{over} ↓	R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5	Mean R_{dup} ↓	Mean R_{over} ↓
RexOmni (Jiang et al., 2026)											
Baseline	NTP	54.90	62.56	58.48	–	–	17.32	21.68	12.69	–	–
FHAR	NTP	63.86	61.66	62.74	4.58%	3.24	23.39	24.99	16.57	58.74%	5.85
FHAR*	NTP	66.94	**68.26**	67.59	2.43%	1.39	36.82	37.92	32.16	35.48%	3.14
FHAR*-GRPO	NTP	66.61	67.77	67.18	2.31%	1.71	37.27	37.24	31.98	35.05%	3.11
WinAR	NTP	**68.32**	67.63	**67.97**	**1.26%**	**0.51**	**45.23**	**38.90**	**39.36**	**13.00%**	**0.87**
Locate Anything (Wang et al., 2026b)											
Baseline	NTP	27.68	46.37	34.66	–	–	8.25	32.41	10.52	–	–
FHAR*	NTP	68.96	**71.52**	**70.34**	0.23%	0.19	53.87	**57.88**	**54.87**	5.90%	0.21
WinAR	NTP	**69.98**	68.08	69.01	**0.01%**	**0.17**	**55.42**	56.22	54.40	**0.49%**	**0.20**
Baseline	PBD	16.45	19.25	17.74	–	–	1.01	10.97	1.54	–	–
FHAR*	PBD	59.98	**69.02**	64.18	0.79%	0.39	23.79	32.47	26.15	14.89%	0.84
WinAR	PBD	**60.60**	68.74	**64.41**	**0.35%**	**0.15**	**23.98**	**34.44**	**27.54**	**2.13%**	**0.02**
Baseline	Hybrid	26.48	36.49	30.69	–	–	6.97	25.39	8.23	–	–
FHAR*	Hybrid	67.95	70.99	69.38	0.57%	0.24	43.98	55.27	49.76	10.60%	0.53
WinAR	Hybrid	**68.44**	**71.37**	**69.87**	**0.29%**	**0.21**	**44.11**	**59.94**	**50.82**	**1.75%**	**0.16**

Table 2: Comparison of inference speed on RSDense and performance on two extremely dense categories.**Boxes/s** denotes the average number of predicted boxes per image per second, reflecting the decoding throughput. **Valid Boxes/s** measures the throughput after removing redundant boxes. Inference speed is measured on eight NVIDIA H100 GPUs.

RexOmni	box / s↑	valid box / s↑	Training Peak GPU Memory (GB)↓	Ship			Small Vehicle		
				R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5	R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5
FHAR*	6.05	3.57	57.62	57.13	47.23	51.71	22.65	17.80	19.94
WinAR	**6.68**	**5.40**	**42.97**	**65.35**	**58.50**	**61.74**	**30.06**	**24.45**	**26.96**

and the over-prediction rate $R_{\text{over}} = (N_{\text{pred}} - N_{\text{gt}})/N_{\text{gt}}$, where N_{dup}, N_{pred}, and N_{gt} denote the numbers of duplicate predictions, total predictions, and ground-truth objects, respectively. R_{dup} measures the fraction of redundant predictions caused by box repetition, while R_{over} measures the relative excess of the total prediction count over the ground-truth count. Both metrics are computed per image and then averaged over the evaluation set, denoted as Mean R_{dup} and Mean R_{over}. More details on dataset settings and evaluation protocols are provided in Sections A.1 and A.3.

4.2 Main Results

Dense Grounding and Box Repetition Suppression. Table 1 reports the main results, where the advantage of WinAR becomes increasingly pronounced as scene density increases. On RexOmni, WinAR improves Recall from 66.94 to 68.32 over the FHAR* baseline on DOTA2 & SODA-A, while reducing Mean R_{dup} from 2.43% to 1.26% and Mean R_{over} from 1.39 to 0.51. The improvement becomes substantially more pronounced on RSDense: WinAR increases Recall from 36.82 to 45.23 and F1 from 32.16 to 39.36, while reducing Mean R_{dup} from 35.48% to 13.00% and Mean R_{over} from 3.14 to 0.87. The same trend is particularly evident for densely populated categories. As shown in Table 2, WinAR improves the Recall of Ship from 57.13 to 65.35 and that of Small Vehicle from 22.65 to 30.06 on RSDense, with their F1 scores increasing from 51.71 to 61.74 and from 19.94 to 26.96, respectively. These improvements are also reflected in the qualitative comparisons in Fig. 1(middle), where WinAR suppresses repeated boxes and recovers substantially more

Table 3: Polygon detection results on DOTA2 and SODA-A. The Baseline is RexOmni model.

Method	DOTA2					SODA-A				
	R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5	Mean R_{dup} ↓	Mean R_{over} ↓	R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5	Mean R_{dup} ↓	Mean R_{over} ↓
Baseline	2.49	3.06	2.44	–	–	1.95	2.27	2.03	–	–
FHAR	46.79	41.82	39.67	8.93%	8.03	47.48	27.19	32.78	7.16%	6.32
FHAR*	51.20	47.98	47.25	6.34%	4.99	54.45	34.71	38.64	3.72%	3.27
WinAR	**52.04**	**49.30**	**48.93**	**2.46%**	**0.89**	**55.35**	**39.38**	**44.00**	**1.93%**	**0.75**

distinct objects. As shown in Fig. 2(b), compressing the historical answer mitigates the progressive concentration of attention on the answer history and maintains more stable conditioning throughout decoding. Consequently, WinAR suppresses the box repetition loop and preserves more of the finite decoding budget for discovering new objects, thereby sustaining object discovery over long decoding horizons. More experimental results across additional datasets are provided in Section A.4.

Generalization across Decoding Paradigms. WinAR also generalizes to the parallel box decoding framework of LocateAnything, as shown in Table 1. Across NTP, PBD, and Hybrid decoding, WinAR consistently suppresses redundant predictions while maintaining comparable or improved grounding accuracy. This effect is particularly evident on the RSDense benchmark. Under NTP, WinAR reduces Mean R_{dup} from 5.90% to 0.49% on RSDense, while achieving comparable F1 (54.87 vs. 54.40) and improving Recall from 53.87 to 55.42. Under PBD, WinAR improves F1 from 26.15 to 27.54, while reducing Mean R_{dup} from 14.89% to 2.13% on RSDense. This demonstrates that WinAR is not tied to token-by-token box generation and can also effectively control historical-answer accumulation when geometric outputs are decoded in parallel.

Generalization to Other Structured Geometric Outputs. Table 3 evaluates whether WinAR generalizes beyond bounding boxes to polygon detection, where each object is represented by a longer sequence of interdependent coordinates (eight coordinate tokens per polygon). Compared with the exposure-matched FHAR* baseline, WinAR improves F1 from 47.25 to 48.93 on DOTA2, while reducing Mean R_{dup} from 6.34% to 2.46% and Mean R_{over} from 4.99 to 0.89. The improvement is more pronounced on SODA-A, which contains generally denser scenes: Recall, Precision, and F1 increase from 54.45, 34.71, and 38.64 to 55.35, 39.38, and 44.00, respectively, while Mean R_{dup} decreases from 3.72% to 1.93% and Mean R_{over} from 3.27 to 0.75. These results demonstrate that WinAR is not specific to fixed-length bounding-box sequences, but also effectively controls historical-answer accumulation when generating more complex structured geometric outputs.

Training and Inference Efficiency. WinAR substantially reduces the memory requirement during training. In our training setup, the peak GPU memory of RexOmni-FHAR* reaches 57.62 GB, whereas RexOmni-WinAR requires only 42.97 GB. This reduction stems from the window-wise training formulation, where completed answer windows are compressed and their raw token-level representations and associated activations can be discarded, substantially reducing the per-GPU memory budget and enabling all WinAR experiments to be conducted on 48 GB NVIDIA A40 GPUs. Table 2 also evaluates the inference efficiency of WinAR on RSDense. WinAR does not consistently improve raw throughput due to the additional overhead of history compression and KV-cache prefilling at window boundaries. However, by substantially reducing redundant predictions, it uses the decoding budget more effectively and increases Valid Boxes/s from 3.57 to 5.40 on RexOmni. Overall, WinAR improves both training scalability and effective inference throughput by replacing ever-growing raw answer histories with compact window-level representations.

4.3 Ablation Study

Compression Configuration. Table 4 studies different compression configurations by varying the window length L, boundary size B, and number of compressed tokens N. Since a smaller window length naturally induces more window-wise training instances, the $L = 100$ configuration receives a larger oversampling weight. To ensure a fair comparison, we apply the same oversampling weight to the $L = 200$ and $L = 300$ configurations, thereby matching their effective sample exposure. We make two main observations. First, more frequent compression consistently suppresses redundant generation, but introduces a trade-off with grounding accuracy. Reducing L from

Table 4: Ablation study of WinAR with different window length L, boundary size B, and number of compressed memory tokens N.

WinAR			DOTA2 & SODA-A					RSDense				
L	B	N	R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5	Mean R_{dup} ↓	Mean R_{over} ↓	R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5	Mean R_{dup} ↓	Mean R_{over} ↓
300	6	8	**68.67**	**68.41**	**68.49**	1.21%	0.48	**48.59**	**42.82**	**44.78**	5.38%	0.66
200	4	6	67.39	67.61	67.22	0.65%	0.42	47.93	42.60	44.56	4.21%	0.57
200	0	20	63.40	67.16	64.82	0.28%	**0.18**	27.96	39.59	32.47	2.59%	**0.04**
100	2	4	68.40	66.15	67.04	**0.12%**	0.27	47.04	41.41	43.71	**0.89%**	0.32

Table 5: Ablation study on the number of layers in the Window History Compressor.

Compressor Layers	DOTA2 & SODA-A					RSDense				
	R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5	Mean R_{dup} ↓	Mean R_{over} ↓	R@IoU↑ 0.5	P@IoU↑ 0.5	F1@IoU↑ 0.5	Mean R_{dup} ↓	Mean R_{over} ↓
8	66.17	63.14	64.62	1.31%	0.66	**45.75**	38.23	39.32	16.87%	1.28
6	**68.32**	**67.63**	**67.97**	1.26%	0.51	45.23	**38.90**	**39.36**	**13.00%**	0.87
4	66.57	64.45	65.49	**1.23%**	**0.44**	43.60	38.20	37.73	15.64%	**0.78**

300 to 100 decreases Mean R_{dup} from 5.38% to 0.89% and Mean R_{over} from 0.66 to 0.32 on RSDense, indicating that more frequent compression more effectively limits the accumulation of raw answer history. However, the corresponding F1 decreases from 44.78 to 43.71, with both Recall and Precision slightly reduced. This suggests that overly short windows, while effective at suppressing repetition, provide less exact local context for subsequent generation. Second, preserving exact boundary tokens is important for maintaining grounding performance. With the same window length $L = 200$, removing boundary preservation ($B = 0$) and increasing the number of compressed tokens to $N = 20$ reduces F1 from 67.22 to 64.82 on DOTA2 & SODA-A and, more substantially, from 44.56 to 32.47 on RSDense compared with $(B, N) = (4, 6)$. This degradation occurs despite the considerably larger compressed-token budget, indicating that simply allocating more latent tokens cannot compensate for the loss of exact local information around window boundaries. Overall, $(L, B, N) = (300, 6, 8)$ achieves the best grounding accuracy while still substantially suppressing redundant generation, and is therefore adopted as our default configuration.

Compressor Depth. Table 5 studies the effect of the depth in the Window History Compressor. A shallow 4-layer compressor is less effective at preserving informative historical context in extremely dense scenes. Increasing the depth to 6 layers improves the overall compression quality, yielding the best F1 on both DOTA2 & SODA-A (67.97) and RSDense (39.36), while also achieving the lowest Mean R_{dup} of 13.00% on RSDense. Further increasing the depth to 8 layers slightly improves Recall on RSDense from 45.23 to 45.75, but decreases Precision from 38.90 to 38.23 and increases Mean R_{dup} from 13.00% to 16.87%, resulting in a slightly lower F1 of 39.32. These results suggest that insufficient depth limits the compressor's capacity to retain useful historical information, whereas additional layers beyond six provide little overall benefit and may introduce unnecessary transformations. We therefore adopt a 6-layer Window History Compressor by default.

5 Conclusions

In this work, we identify a critical long-horizon failure mode of MLLM-based detectors in dense vision-language grounding, where generation progressively collapses into a self-reinforcing box repetition loop. To address this issue, we propose **Compressed Window Auto-Regression (WinAR)**, which preserves exact autoregression within the current window while representing completed windows with compact latent tokens, thereby limiting repeated exposure to raw historical coordinates without discarding cross-window dependencies. Experiments show that WinAR increasingly benefits denser scenes. WinAR further generalizes across decoding paradigms and structured geometric outputs, while improving effective detection efficiency and reducing training memory for long sequences. These results establish compressed window autoregression as a practical paradigm for reliable long-horizon vision-language grounding.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language polishing of the manuscript and with debugging implementation code. All AI-assisted text was carefully reviewed and revised by the authors to ensure accuracy, clarity, and consistency with the intended scientific claims. For code-related assistance, AI-generated suggestions were independently inspected, verified, and tested by the authors before being incorporated into our implementation. We take full responsibility for the final content of this work, including all text, claims, experimental results, and code produced with the aid of generative AI.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. 3

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6154–6162, 2018. 3

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*, pp. 213–229. Springer, 2020. 3

Ting Chen, Saurabh Saxena, Lala Li, David J Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection. *arXiv preprint arXiv:2109.10852*, 2021. 3

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. *arXiv preprint arXiv:2312.14238*, 2023. 3

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. 2, 6, 13

Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. Yolo-world: Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16901–16911, 2024. 1

Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7373–7382, June 2021. 3

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019. 3

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3, 4

Eran Goldman, Roei Herzig, Aviv Eisenschtat, Jacob Goldberger, and Tal Hassner. Precise detection in densely packed scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5227–5236, 2019. 1, 2, 13, 14

Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, Haoqi Fan, Jian Wang, Jianyu Jiang, Jiawei Wang, et al. Seed1. 5-vl technical report. *arXiv preprint arXiv:2505.07062*, 2025. 3

Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25472–25483, 2026. 1, 3, 4, 6, 7, 14, 15

Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10965–10975, 2022. 3

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. 3

Yuxuan Li, Yicheng Zhang, Wenhao Tang, Yimian Dai, Ming-Ming Cheng, Xiang Li, and Jian Yang. Visual instruction pretraining for domain-specific foundation models. *arXiv preprint arXiv:2509.17562*, 2025. 3

Yuxuan Li, Yuming Chen, Yunheng Li, Ming-Ming Cheng, Xiang Li, and Jian Yang. Unifying heterogeneous multi-modal remote sensing detection via language-pivoted pretraining. In *ICML*, 2026. 3

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*, pp. 740–755. Springer, 2014. 2, 13, 14

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017a. 3

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*, pp. 2980–2988, 2017b. 3

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pp. 38–55. Springer, 2024. 1, 3

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *Advances in Neural Information Processing Systems*, 38:100092–100118, 2026. 2

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021. 3

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016. 3

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. 3

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025. 3

Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9627–9636, 2019. 3

Chenxu Wang, Chunyan Xu, Xiang Li, YuXuan Li, Xu Guo, Ziqi Gu, and Zhen Cui. Multi-clue consistency learning to bridge gaps between general and oriented object in semi-supervised detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7582–7590, 2025a. 3

Chenxu Wang, Yuxuan Li, Yunheng Li, Xiang Li, Jingyuan Xia, and Qibin Hou. Slip-rs: Structured-attribute language-image pre-training for remote sensing object detection. *arXiv preprint arXiv:2605.23144*, 2026a. 1

Shihao Wang, Shilong Liu, Yuanguo Kuang, Xinyu Wei, Yangzhou Liu, Zhiqi Li, Yunze Man, Guo Chen, Andrew Tao, Guilin Liu, et al. Locateanything: Fast and high-quality vision-language grounding with parallel box decoding. *arXiv preprint arXiv:2605.27365*, 2026b. 1, 2, 3, 4, 7, 14

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025b. 3

Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3974–3983, 2018. 1, 2, 6, 13, 15

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024. 2

Tao Zewei and Huang Yunpeng. Magiattention: A distributed attention towards linear scalability for ultra-long context, heterogeneous mask training, 2025. 2

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*, pp. 9759–9768, 2020. 3

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

A APPENDIX

APPENDIX CONTENTS

- **Benchmark Details** summarizes the construction, preprocessing, splits, and statistics of DOTA2 & SODA-A, RSDense, DOTA2-Poly, SODA-A-Poly, SKU110K, and COCO.
- **Additional Implementation Details** provides the training configurations for RexOmni and LocateAnything.
- **Evaluation Protocol** describes the capped evaluation protocol and the complementary all-prediction evaluation for analyzing object discovery, prediction redundancy, and generation termination.
- **More Experiments** reports all-prediction comparisons across COCO, DOTA2 & SODA-A, SKU110K, and RSDense.

A.1 BENCHMARK DETAILS

In this section, we summarize the datasets and benchmark settings used throughout our experiments. We evaluate WinAR from three complementary perspectives. First, we use DOTA2 & SODA-A (Xia et al., 2018; Cheng et al., 2023), RSDense, and SKU110K (Goldman et al., 2019) to assess performance in dense and extremely dense scenes across both remote-sensing and general visual domains. Second, we construct DOTA2-Poly (Xia et al., 2018) and SODA-A-Poly (Cheng et al., 2023) to evaluate long-horizon structured generation with quadrilateral outputs, where each object requires twice as many coordinate tokens as a standard bounding box. Finally, we use COCO (Lin et al., 2014) to verify that the improvements of WinAR in dense scenes do not compromise performance in ordinary visual scenarios.

- **DOTA2 & SODA-A:** DOTA2 (Xia et al., 2018) and SODA-A (Cheng et al., 2023) are both remote-sensing object detection datasets with closely aligned scene domains and a substantial number of densely populated images. We therefore combine the two datasets to construct a unified training and evaluation benchmark for studying dense vision-language grounding across different scene densities. For DOTA-v2.0, we follow the official single-scale preprocessing and evaluation protocol. Specifically, the original high-resolution images are cropped into 1024×1024 patches with an overlap of 200 pixels between adjacent patches. The resulting training split, containing 13,447 images, is included in the unified training set, while the validation split, containing 4,053 images, is used as part of the unified test set. For SODA-A, we follow the official preprocessing setting and crop the original images into 800×800 patches with a stride of 650 pixels. We merge its training and validation splits, yielding 32,181 images for training, while the test split, containing 14,654 images, is included in the unified test set. After preprocessing, the DOTA2 and SODA-A training data are jointly used for model training, while the DOTA2 validation split and the SODA-A test split are combined to form the DOTA2 & SODA-A evaluation benchmark used throughout this work.
- **RSDense:** To specifically evaluate WinAR in highly dense visual scenes, we construct RSDense from the DOTA2 (Xia et al., 2018) & SODA-A (Cheng et al., 2023) evaluation set by selecting images containing more than 200 objects. RSDense consists of 712 images, with an average of 412 objects per image, providing a challenging benchmark for long-horizon dense grounding. Among all categories, *Ship* and *Small Vehicle* exhibit the highest object densities and therefore serve as two representative dense categories in our analysis.
- **DOTA2-Poly and SODA-A-Poly:** We evaluate WinAR on the oriented-box annotations of DOTA2 (Xia et al., 2018) and SODA-A (Cheng et al., 2023) to assess its generalization to quadrilateral geometric outputs. Unlike a standard horizontal box, which is represented by four coordinate tokens, an oriented box is described by four vertices and therefore requires eight coordinate tokens. For the same number of objects, this doubles the output length and substantially extends the decoding horizon, imposing greater demands on long-range autoregressive modeling and historical-context management. This setting therefore provides a more challenging test of WinAR under longer and more complex structured generation. In these experiments, we train and evaluate on the two datasets separately rather than combining them. For DOTA2-Poly, we use the oriented-box annotations of DOTA2, train on

its training split, and evaluate on the corresponding validation split. For SODA-A-Poly, we use the oriented-box annotations of SODA-A, merge its training and validation splits for training, and evaluate on the test split. All other preprocessing and training settings are kept consistent with their horizontal-box counterparts, with only the output representation changed from 4-token box coordinates to 8-token quadrilateral coordinates.

- **SKU110K:** The SKU110K dataset (Goldman et al., 2019) is a large-scale object detection dataset focused on densely packed retail products on supermarket shelves, representing a challenging dense-scene setting in general visual domains. We use its training split, containing 8,769 images, for model training and evaluate on the test split, which contains 2,920 images. This benchmark is used to assess whether WinAR generalizes beyond remote-sensing imagery to dense object detection in every scenes.
- **COCO:** To verify that the gains of WinAR in dense scenes do not come at the expense of performance in ordinary visual scenarios, we further evaluate it on the widely used COCO (Lin et al., 2014) benchmark. Following standard practice (Jiang et al., 2026), we train on the COCO training set and evaluate on the validation set. This setting allows us to assess whether WinAR preserves competitive grounding performance under typical scene densities while improving robustness in highly dense scenes.

Table 6: Detailed statistics of the training datasets used in our experiments.

Dataset	#Images	#Annotations	Min / Image	Max / Image	Avg. / Image
DOTA2 & SODA-A	45,628	1,493,991	1	2,831	32.74
DOTA2-Poly	13,447	473,548	1	2,831	35.22
SODA-A-Poly	32,181	1,020,443	1	2,653	31.71
SKU110K	8,769	1,293,724	1	718	147.53
COCO	117,266	860,001	1	93	7.33

Table 7: Detailed statistics of the test datasets used in our experiments.

Dataset	#Images	#Annotations	Min Objects/Image	Max Objects/Image	Avg. Objects/Image
DOTA2 & SODA-A	18,707	656,706	1	2,305	35.14
RSDense	712	293,371	201	2,305	412.04
SKU110K	2,920	429,411	20	533	147.06
DOTA2-Poly	4,053	140,103	1	2,307	34.57
SODA-A-Poly	14,654	517,099	1	1,692	35.29
COCO	4,952	36,781	1	63	7.43

A.2 Additional Implementation Details

Table 8 summarizes the training configurations for RexOmni (Jiang et al., 2026) and LocateAnything (Wang et al., 2026b). A key consideration in our comparison is that window-wise training in WinAR naturally increases both the number of optimizer updates and the effective exposure of long-answer samples. To disentangle the benefit of WinAR from that of additional training exposure, we carefully construct three training settings:

- **RexOmni-FHAR.**We first reproduce the original Full-History Auto-Regression (FHAR) training protocol of RexOmni. The model is trained for 4 epochs with a per-device batch size of 1 and 16 gradient-accumulation steps, resulting in 1,424 optimizer updates. We use AdamW with a base learning rate of 4×10^{-5}, an MM-projector learning rate of 4×10^{-5}, and a vision-tower learning rate of 4×10^{-6}. The learning rate follows a cosine schedule with a warmup ratio of 0.05. We set the weight decay to 0.05, the maximum gradient norm to 1.0, and the maximum sequence length to 20,000 tokens. Each training sample is exposed once per epoch, regardless of its answer length.
- **RexOmni-WinAR.**RexOmni-WinAR follows the same optimization settings as RexOmni-FHAR, while additionally optimizing the Window History Compressor with a learning

rate of 4×10^{-4}. WinAR adopts a window size of $L = 300$ and decomposes each long answer into multiple consecutive windows that are optimized separately. Consequently, long-answer samples receive proportionally more training exposure than under conventional FHAR. Specifically, a sample containing T answer tokens contributes approximately $\lceil T/L \rceil$ window-level training instances; for example, an answer of 3,000 tokens is divided into 10 windows when $L = 300$. This window-wise formulation therefore implicitly oversamples long samples and increases the number of training iterations within each epoch. With our training set, RexOmni-WinAR performs 1,864 optimizer updates over 4 epochs.

- **RexOmni-FHAR*.** To isolate the effect of compressed window autoregression from the additional sample exposure introduced by window-wise training, we construct an exposure-matched FHAR* baseline. FHAR* retains the original full-history autoregressive training objective, but modifies data sampling such that a sample with T answer tokens is repeated approximately $\lceil T/L \rceil$ times, matching the effective exposure assigned to that sample by RexOmni-WinAR. This results in the same 1,864 optimizer updates as WinAR over 4 epochs, while preserving the FHAR decoding context during training. All remaining optimization hyperparameters are kept identical to RexOmni-FHAR.
- **RexOmni-FHAR*-GRPO.** Post-training with GRPO is an important component of RexOmni for improving detection quality. In our experiments, GRPO is applied to the RexOmni baseline for comparison with the original RexOmni training recipe. We initialize the policy from the SFT checkpoint of RexOmni-FHAR* and perform GRPO post-training for one epoch on the subset of the joint DOTA2 and SODA-A training set containing more than 200 objects per image. Training is distributed over 8 NVIDIA H100 GPUs. For each rollout iteration, we sample a batch of 8 prompts and generate 4 rollouts for each prompt. The four candidate trajectories associated with the same prompt form a response group for group-relative reward comparison in GRPO, resulting in 32 sampled responses per rollout batch. The actor is optimized with a global batch size of 8.
- **LocateAnything-FHAR*.** We follow the same training strategy as RexOmni-FHAR*. To account for the increased sample exposure introduced by window-wise training in LocateAnything-WinAR, we oversample the corresponding training samples to match its effective sample exposure and number of training iterations. All other optimization hyperparameters are kept consistent with those used for RexOmni.
- **LocateAnything-WinAR.** LocateAnything-WinAR follows the same training strategy as RexOmni-WinAR, except that windows are constructed over box-aligned atomic blocks rather than individual tokens. Specifically, each long answer is partitioned into consecutive block-level windows, which generates additional training instances and consequently increases both the number of iterations per epoch and the exposure frequency of samples spanning multiple windows. The remaining optimization settings, including the learning rate of the Window History Compressor, are kept consistent with RexOmni-WinAR.

Therefore, FHAR* matches WinAR in both the total number of optimizer updates and the effective exposure of long-answer samples, while differing only in how the historical answer is represented during autoregressive training. This controlled comparison allows us to distinguish the effect of WinAR from gains that may arise simply from increased training exposure.

A.3 Evaluation Protocol

In the main paper, we follow the protocol adopted by RexOmni (Jiang et al., 2026) and use **Recall**, **Precision**, and **F1 score** as the primary evaluation metrics at an IoU threshold of 0.5. We retain the top 100 predictions for each predicted category in each image. For each category, predictions from all evaluation images are then aggregated and matched against the corresponding ground-truth boxes to compute Recall and Precision, from which the F1 score is obtained as their harmonic mean. The final Recall, Precision, and F1 are averaged across categories. For DOTA2 & SODA-A, we adopt the same category-wise evaluation procedure, but follow the standard DOTA2 (Xia et al., 2018) setting and retain up to 1,000 predictions per category for each image. For RSDense, however, the severe category imbalance makes category-averaged evaluation less representative of performance in extremely dense scenes. We therefore evaluate all predicted boxes and perform class-agnostic matching within each image. Recall, Precision, and F1 score are computed independently for each

Table 8: Training hyperparameters for RexOmni and LocateAnything. RO and LA denote RexOmni and LocateAnything, respectively.

Hyperparameter	**RO-FHAR**	**RO-FHAR***	**RO-WinAR**	**LA-FHAR***	**LA-WinAR**
Training Epochs	4	4	4	4	4
Oversample	False	True	True	True	True
Optimize Steps	1,424	1,864	1,864	2,004	2,004
Per-device Batch Size	1	1	1	1	1
Gradient Accumulation Steps	16	16	16	16	16
Base Learning Rate	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}
MM Projector Learning Rate	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}
Vision Tower Learning Rate	4×10^{-6}	4×10^{-6}	4×10^{-6}	4×10^{-6}	4×10^{-6}
Window History Compressor LR	–	–	4×10^{-4}	–	4×10^{-4}
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
LR Schedule	Cosine	Cosine	Cosine	Cosine	Cosine
Warmup Ratio	0.05	0.05	0.05	0.05	0.05
Weight Decay	0.05	0.05	0.05	0.05	0.05
Max Gradient Norm	1.0	1.0	1.0	1.0	1.0
Max Sequence Length	20,000	20,000	20,000	20,000	20,000
Trainng NVIDIA GPUs(8)	H100(80GB)	H100(80GB)	A40(48GB)	H100(80GB)	A40(48GB)

image and then averaged over all images in RSDense. The evaluation prompt contains only the ground-truth categories present in each image.

Capped evaluation alone does not fully characterize the generation behavior of autoregressive detectors. For conventional detectors, predictions beyond the top-K limit are typically removed only during evaluation, whereas every prediction generated by an autoregressive detector has already consumed decoding steps and computational budget. This distinction is particularly important in dense scenes, where a model may enter a box repetition loop and continue producing large numbers of redundant predictions after most objects have already been covered. Under top-K evaluation, many of these late predictions can be excluded from the final metrics, obscuring the actual redundancy and termination behavior of the model. The truncation may also alter the observed Recall–Precision trade-off: valid detections beyond the cap can be discarded, limiting Recall, while redundant late predictions can be filtered out at the same time, potentially yielding an overly optimistic Precision relative to the model's complete output. Therefore, although capped evaluation is useful for maintaining comparability with existing benchmarks, it provides only a partial view of long-horizon autoregressive detection, where the ability to terminate generation after sufficiently covering the scene is itself an important aspect of both detection quality and inference efficiency.

To complement the standard protocol, we further report an **all-prediction evaluation** in the supplementary material, where all predictions generated by the model are retained without imposing an additional per-image prediction cap. We otherwise follow the same category-wise evaluation procedure: predictions of each category are aggregated across all evaluation images to compute Recall and Precision, F1 is obtained as their harmonic mean, and the final metrics are averaged across categories. This protocol is not intended to replace the standard evaluation, but rather to provide a more complete characterization of **object discovery**, **prediction redundancy**, and **generation termination behavior**. This evaluation is particularly relevant to WinAR, whose objective is not only to improve recall in dense scenes, but also to suppress the box repetition loop and allocate the finite decoding budget more effectively to discovering new objects.

A.4 More Experiments

Additional Results on More Datasets. Table 9 reports the results under the all-prediction evaluation setting, where every generated prediction is included without imposing a per-image prediction cap. WinAR consistently improves F1 over the exposure-matched FHAR* baseline across all evaluated datasets, while the gains become substantially larger as scene density increases. On COCO, WinAR preserves performance in ordinary scenes, improving F1 from 65.48 to 66.04. Similar improvements are observed on DOTA2 & SODA-A and SKU110K, where F1 increases by 0.75 and

Table 9: Performance comparison between RexOmni-FHAR* and RexOmni-WinAR on COCO, DOTA2 & SODA-A, RSDense, and SKU110K under the all-prediction evaluation setting. RexOmni-FHAR* denotes the RexOmni-FHAR baseline trained with long-sample oversampling.

Dataset	Method	Precision@0.50↑	Recall@0.50↑	F1@0.50↑
COCO	FHAR*	63.28	67.84	65.48
	WinAR	64.31 (+1.03)	67.87 (+0.03)	66.04 (+0.56)
DOTA2 & SODA-A	FHAR*	62.08	66.94	64.42
	WinAR	62.79 (+0.71)	68.40 (+1.46)	65.17 (+0.75)
SKU110K	FHAR*	53.89	81.75	64.96
	WinAR	54.93 (+1.04)	80.94 (−0.81)	65.44 (+0.48)
RSDense	FHAR*	55.90	38.95	45.91
	WinAR	53.36 (−2.54)	47.69 (+8.74)	50.37 (+4.46)

Table 10: Comparison of inference throughput and training peak GPU memory. **Boxes/s** denotes the average number of predicted boxes per image per second, while **Valid Boxes/s** measures the throughput after removing redundant predictions. Inference speed is measured on eight NVIDIA H100 GPUs.

Method	Mode	Boxes/s↑	Valid Boxes/s↑	Training Peak GPU Memory (GB)↓
RexOmni–FHAR*	NTP	6.05	3.57	57.62
RexOmni-WinAR	NTP	**6.68**	**5.40**	**42.97**
Locate Anything–FHAR*	NTP	**5.46**	5.14	67.38
Locate Anything-WinAR	NTP	5.40	**5.37**	**44.43**
Locate Anything-FHAR*	PBD	**29.03**	24.63	67.38
Locate Anything-WinAR	PBD	26.15	**25.61**	**44.43**

0.48 points, respectively. The advantage is most pronounced on RSDense: WinAR improves Recall from 38.95 to 47.69 (+8.74) and F1 from 45.91 to 50.37 (+4.46), despite a moderate decrease in Precision. These results further show that the benefit of WinAR is not an artifact of prediction truncation during evaluation. Instead, by suppressing redundant long-horizon generation, WinAR enables the autoregressive detector to allocate more of its decoding budget to previously uncovered objects, yielding increasingly pronounced gains in dense scenes while maintaining performance under ordinary scene densities.

Training and Inference Efficiency. Table 10 evaluates the training memory and inference efficiency of WinAR. WinAR substantially reduces peak GPU memory during training. On RexOmni, peak memory decreases from 57.62 GB with FHAR* to 42.97 GB with WinAR, corresponding to a 25.4% reduction. The reduction is even larger on LocateAnything, from 67.38 GB to 44.43 GB (34.1%). This benefit stems from the window-wise training formulation: once a completed answer window is compressed, its raw token-level representations and associated activations can be discarded, substantially reducing the per-GPU memory requirement and allowing WinAR to fit within the 48 GB memory budget of NVIDIA A40 GPUs. At inference time, WinAR does not consistently improve raw prediction throughput because history compression and KV-cache prefilling introduce additional overhead at window boundaries. Nevertheless, WinAR consistently improves effective throughput by suppressing redundant predictions. On RexOmni, Valid Boxes/s increases from 3.57 to 5.40. For LocateAnything, Valid Boxes/s improves from 5.14 to 5.37 under NTP and from 24.63 to 25.61 under PBD, despite slight decreases in raw Boxes/s. Overall, WinAR substantially improves training scalability while allocating a larger fraction of the decoding budget to non-redundant predictions during inference.

Qualitative Comparison. The visualization results further demonstrate the effectiveness of WinAR in long-horizon dense grounding. Compared with FHAR*, WinAR produces substantially fewer repeated predictions and recovers more distinct objects in crowded scenes. For polygon detection,

WinAR also yields more complete and spatially consistent predictions, indicating that its benefit extends beyond axis-aligned boxes to more general geometric outputs.

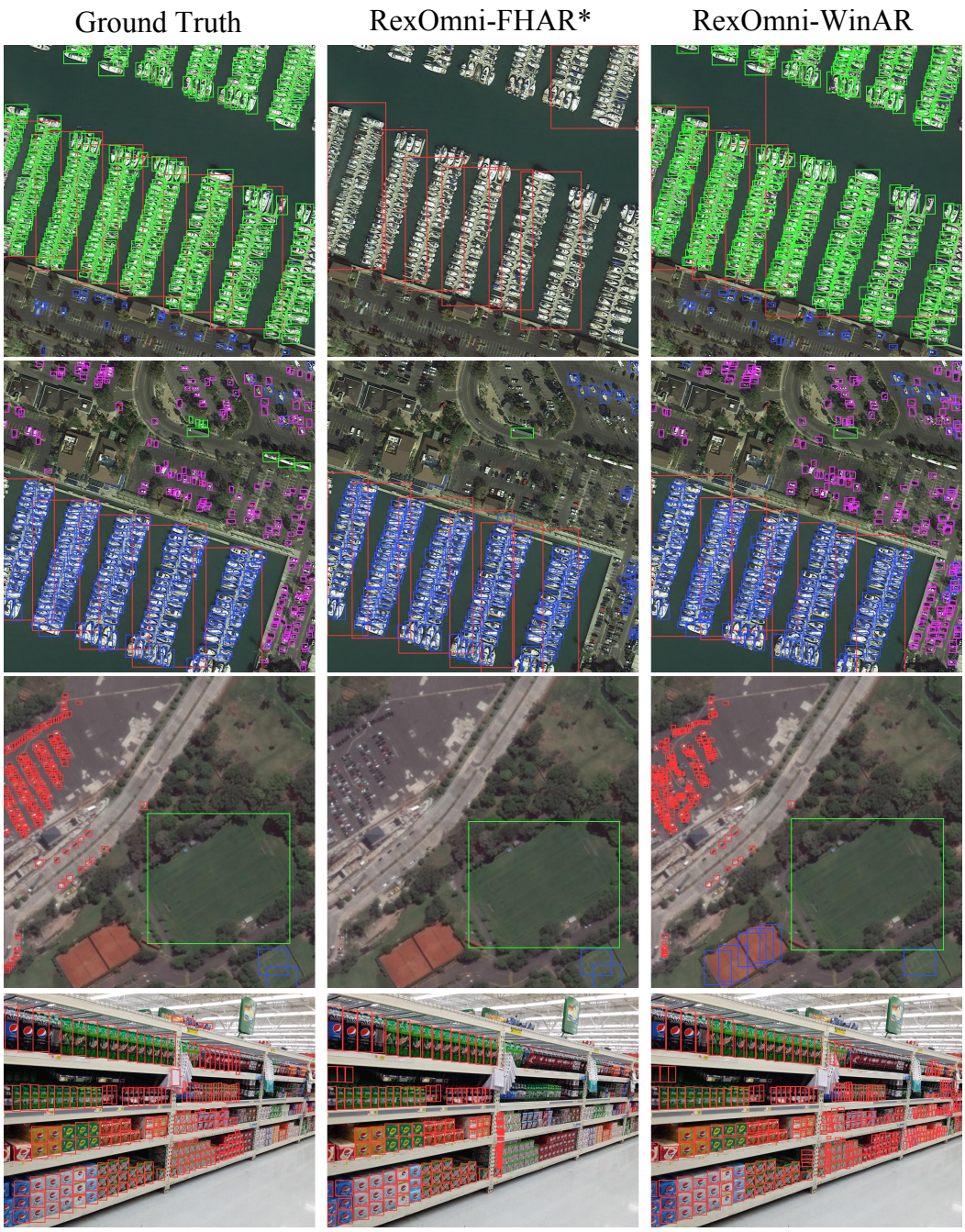


Figure 5: Visualization Comparison in Dense Scenes.



Figure 6: Visualization Comparison in Dense Scenes.

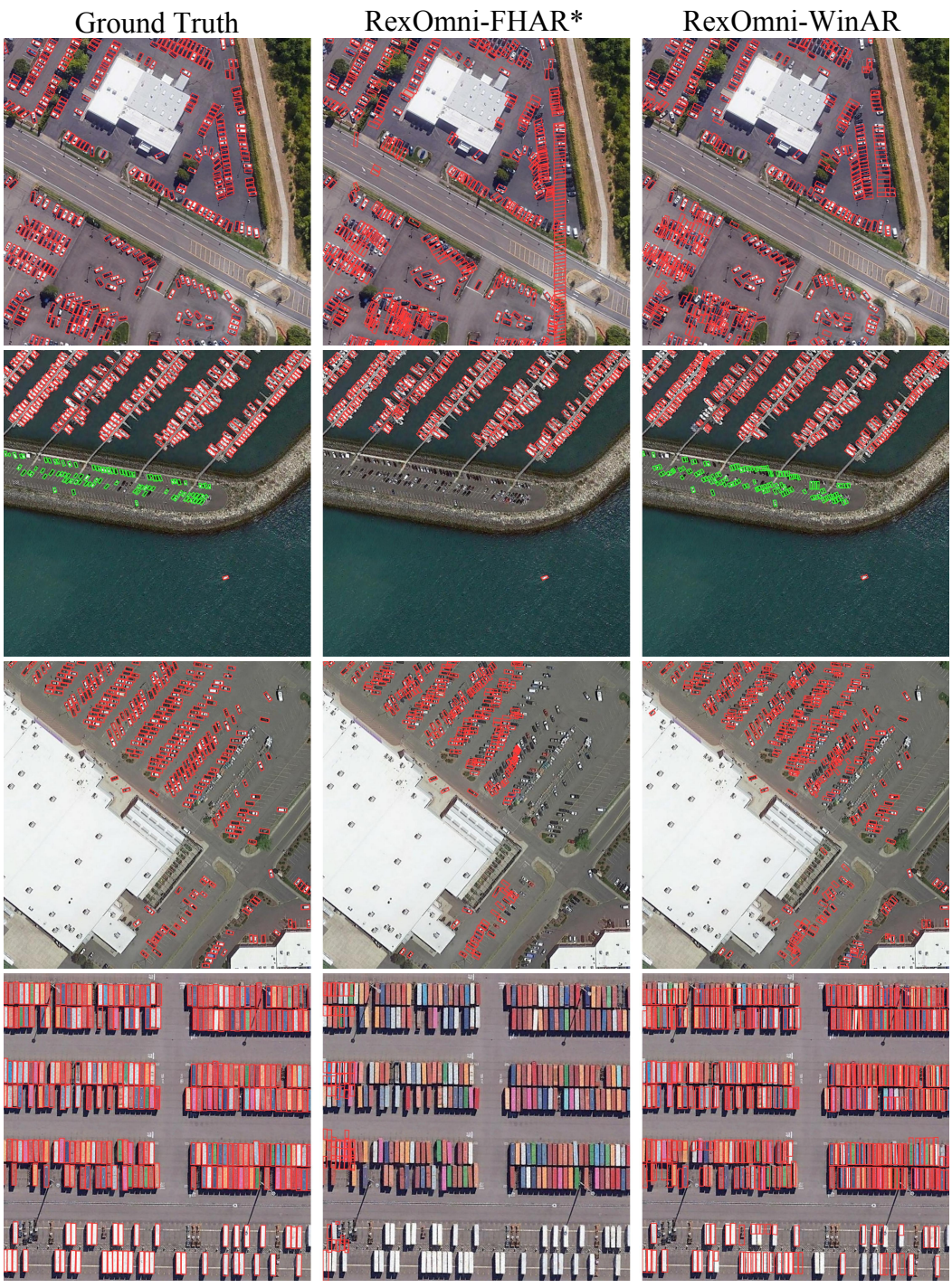


Figure 7: Visualization Comparison in Polygon Detection.

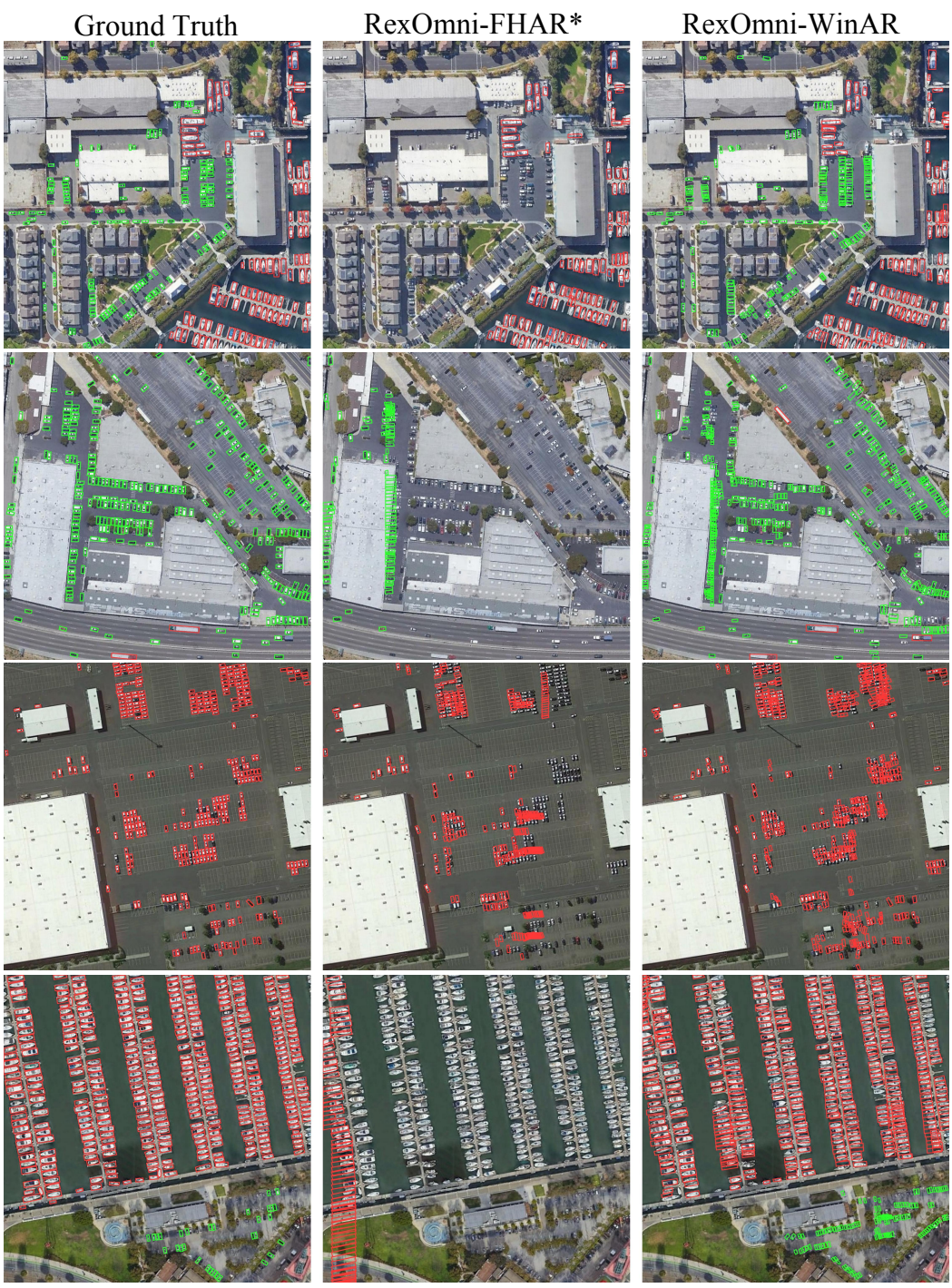


Figure 8: Visualization Comparison in Polygon Detection.