DCAGENT: TOWARDS DENSE VISUAL PERCEPTION VIA A DIVIDE-AND-CONQUER TOOL-USE AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Frontier Vision-Language Models (VLMs) collapse on dense visual perception—tasks that require enumerating or localising many small, tightly packed objects. The structural cause is a fixed-budget visual encoder that smears dense objects before they reach the language model. The intuitive remedy—divide-and-conquer crops—has been explored as a hard-coded external pipeline, but not as a learned, agentic capability inside the VLM itself. We present DCAGENT, a tool-use agent that learns to divide and conquer its own image input through a compact visual interface: `Divide` splits the current view into agent-selected sub-pieces, `Detect` emits leaf detections, and `Merge` aggregates them in the original frame with class-aware non-maximum suppression. The multi-turn tool-use policy is trained with GRPO under an end-to-end final reward combining count MAE and DETR-style Hungarian-matched detection IoU on the merged output. Three task-specialised agents are trained on **SKU-110K**, **SOD-Aerial**, and **LVLM-Count** from a Qwen3-VL-8B starting point, then consolidated into a single deployable model with four-teacher Online Policy Distillation, using the Qwen3-VL-8B base as an anchor teacher. The merged model internalises the divide-and-merge policy rather than the patterns of any single dataset: it auto-triggers tool rollouts on counting and dense-perception inputs, bypasses them on knowledge-and-reasoning inputs, improves all three training datasets, generalises to held-out CountBench, and preserves the Qwen3-VL-8B base on mainstream VLM benchmarks.

1 INTRODUCTION

Modern Vision-Language Models (VLMs) still struggle with a basic visual operation: enumerating and localising dense objects. The failure is not just a matter of prompt wording. A fixed-size visual encoder must compress arbitrarily high-resolution input into a finite token budget, so small, tightly packed objects are blurred before they reach the language model (Deitke et al., 2025; Wang et al., 2024a; Paiss et al., 2023). Table 1 shows this gap across retail shelves, aerial imagery, and cluttered counting.

From external scaffold to internal tool-use agent. A natural remedy is *divide-and-conquer* (D&C): crop the image into sub-tiles, solve each view, then aggregate. Recent training-free pipelines—LVLM-Count (Ghofrani et al., 2025) and DC^2 (Wang et al., 2024b)—wrap a frozen VLM with fixed grids and heuristic count summing. This external scaffold is data-agnostic: it wastes calls on empty regions, fragments boundary objects, and then double-counts the fragments it created. More importantly, the VLM never learns *when* to split, *where* to recurse, or *how* to merge partial predictions.

Modern LLM agents have shown that complex behaviour can be learned by letting the model *call tools*—`python` (Li et al., 2025; Schick et al., 2023), `search` (Guo et al., 2025; Yao et al., 2023), GUI actions (Lu et al., 2025; Wang et al., 2025a; Yang et al., 2026; Qin et al., 2025), and general API use (Qin et al., 2024; Yang et al., 2023)—inside a multi-turn RL loop. We treat dense visual perception the same way: splitting is a tool that operates on the image itself. DCAGENT exposes three structured visual calls (Fig. 3):

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.

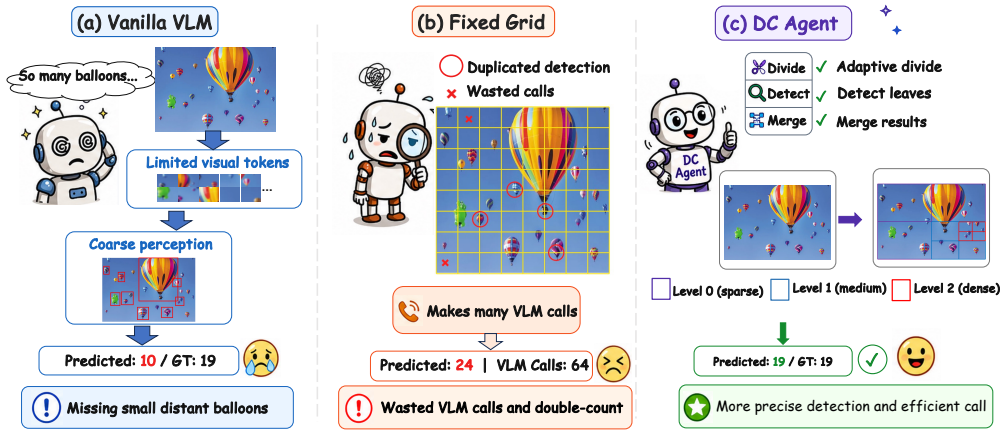
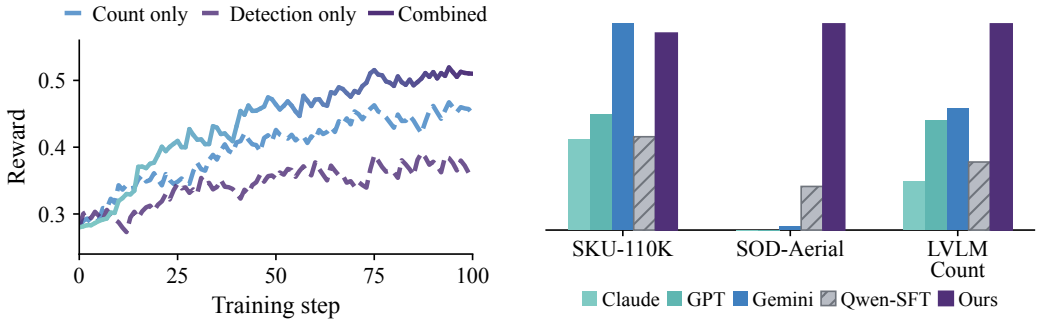


Figure 1: **Dense perception under a tight visual-token budget.** A vanilla VLM sees only a coarse view, while a fixed-grid divide-and-conquer wrapper spends calls uniformly and creates boundary duplicates. DCAGENT instead learns an adaptive quadtree of `Divide`, `Detect`, and `Merge` calls, spending resolution on dense regions and suppressing duplicates at merge time.



(a) Joint reward training yields higher final reward.

(b) Relative dense-perception gains.

Figure 2: **Training signal and dense-perception gains.** (a) Count-only and detection-only rewards improve the policy but saturate lower than their joint objective. (b) Relative per-task scores (normalised within each dataset; higher is better) summarise the dense-perception gains reported in Tab. 1.

- `Divide`: crops the current view into agent-selected quadrants, with overlap and child cores for boundary handling.
- `Detect`: terminates a branch by predicting local centres and boxes on the current view.
- `Merge`: maps all leaf predictions back to the original frame, applies core ownership and class-aware NMS, and emits the final count and detection set.

Repeated `Divide` calls form an adaptive quadtree, and `Merge` replaces brittle count summing with explicit duplicate suppression.

Multi-turn tool-use RL with bipartite-matching reward. Training a multi-turn tool-use agent for set-prediction is non-trivial: the leaf outputs are unordered sets of bounding boxes, and naive cross-entropy supervision breaks under permutation. We address this with an end-to-end final reward combining (i) a count-MAE term and (ii) DETR-style Hungarian-matched detection IoU (Carion et al., 2020) computed on the global `Merged` set. The combination is not only a bookkeeping convenience: count rewards penalise missing or duplicate instances, while detection rewards favour spatially faithful leaves, and the two signals empirically reinforce each other during GRPO (Fig. 2a). No process or step-level shaping is added—the policy is trained purely from the final reward, with the bounded recursion depth $D_{\max}$ providing the only structural constraint.

Four-teacher OPD and task-conditional tool use. Training one specialist per benchmark is effective but awkward to deploy, and direct SFT on dense annotations erodes general VLM ability. We therefore consolidate the specialists with four-teacher OPD (*Online Policy Distillation*): three RL specialists teach their target distributions, while the original Qwen3-VL-8B base acts as an anchor teacher for knowledge-and-reasoning inputs. Routing each calibration input to its owning teacher yields a task-conditional student policy: recurse on dense/counting inputs, answer in one turn on ordinary VLM inputs. The single 8B checkpoint improves dense perception and transfers to held-out CountBench while largely preserving mainstream VLM ability (Tab. 1).

Contributions.

1. **Agentic divide-and-conquer.** We turn fixed-grid D&C into a learned multi-turn VLM policy over `Divide`, `Detect`, and `Merge`.
2. **End-to-end tool-use RL.** A final count/detection reward with Hungarian-matched IoU trains stable recursive behaviour under GRPO without step-level shaping.
3. **Four-teacher consolidation.** OPD merges three dense-perception specialists with the Qwen3-VL-8B base anchor into a single task-conditional checkpoint.

2 Related Work

Dense counting and detection in VLMs. Modern VLMs handle small object counts via subitising (Paiss et al., 2023) but degrade sharply as density rises (Deitke et al., 2025). Specialised counting architectures—density regressors (Ranjan et al., 2021; Zhang et al., 2016), exemplar-based detectors (Nguyen et al., 2022), or open-vocabulary models such as CountGD (Amini et al., 2024)—are accurate in narrow domains but lack the language flexibility of generalist VLMs. We study the gap on SKU-110K (Goldman et al., 2019), SOD-Aerial (Zhao et al., 2023), and LVLM-Count (Ghofrani et al., 2025).

Divide-and-conquer for high-resolution VLMs. Crop-and-stitch is well-studied in detection (Ünel et al., 2019; Akyon et al., 2022) and has recently been adapted to VLMs: LVLM-Count (Ghofrani et al., 2025) prepends a Grounded-SAM (Ren et al., 2024) crop pipeline; DC^2 (Wang et al., 2024b) fixes a uniform grid with patch descriptions and hallucination filtering. Both treat splitting as a hard-coded scaffold with heuristic count summing. DCAgent instead *learns* whether and where to recurse end-to-end through RL, and merges with class-aware NMS over leaf detections.

Multi-turn tool-use RL and detection-style rewards. RL-from-tool-use is now standard in code (Li et al., 2025), search (Guo et al., 2025), web browsing (Qi et al., 2024; Yang et al., 2026), and GUI control (Lu et al., 2025; Wang et al., 2025a), with GRPO (Shao et al., 2024) or PPO under verifiable rewards. Visual perception has been a holdout: V* (Wu & Xie, 2024) and SEAL (Wu et al., 2024) use search-then-look heuristics, while VisualPRM (Wang et al., 2025b) and VLM-R1 (Shen et al., 2025) apply RL to single-turn visual reasoning. DCAgent treats splitting as a visual tool that operates on the image itself, and trains multi-turn rollouts with a DETR-style Hungarian-matched IoU reward (Carion et al., 2020; Zhu et al., 2021).

Policy distillation and model merging. Independently fine-tuned experts are often consolidated by averaging, Fisher merging, TIES, DARE, Model Stock, task arithmetic, permutation alignment, or weight-ensembling MoE (Wortsman et al., 2022; Matena & Raffel, 2022; Yadav et al., 2023; Yu et al., 2024; Jang et al., 2024; Ilharco et al., 2023; Stoica et al., 2024; Tang et al., 2024). These are weight-level operations with no explicit notion of which inputs should activate each capability. OPD instead follows policy distillation (Hinton et al., 2015; Rusu et al., 2016; Czarnecki et al., 2019; Ross et al., 2011): each specialist is an action-distribution teacher, and the base model anchors knowledge-and-reasoning inputs through routed on-policy KL.

3 Method

We present DCAgent in three parts: the adaptive visual tool interface (§3.1), the multi-turn GRPO objective (§3.2), and the four-teacher OPD procedure that merges three dense-perception specialists with the original Qwen3-VL-8B base anchor (§3.3). Figure 3 summarises the tool-call interface and training loop.

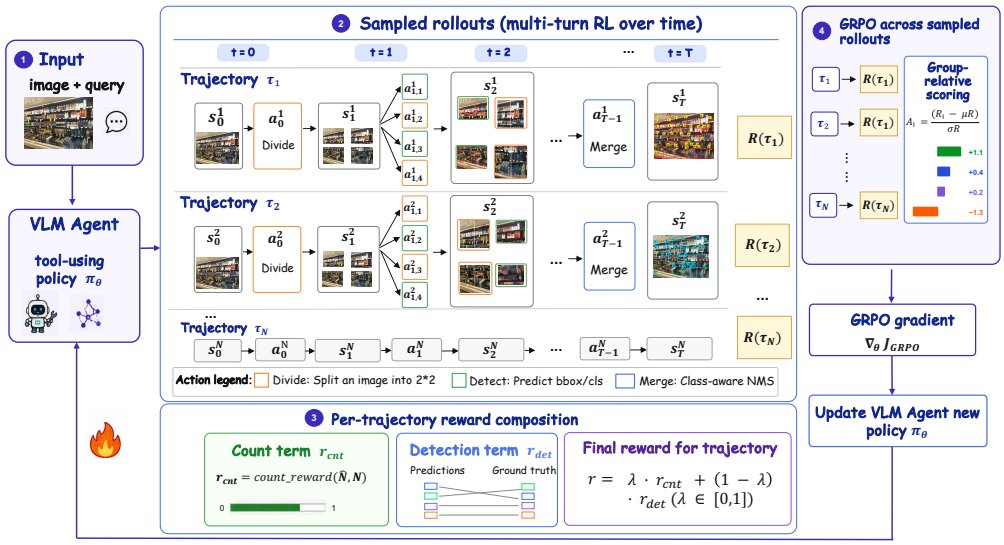


Figure 3: **Overview of DCAGENT.** A rollout traverses a quadtree of structured calls: `Divide` creates child views, `Detect` emits leaf predictions, and `Merge` maps leaves back to the original frame with class-aware NMS. Each completed rollout receives a single final reward combining count and detection quality; GRPO broadcasts the group-relative advantage to all learnable action tokens in the trajectory.

3.1 THE DCAGENT TOOL-USE AGENT

Three visual tools. We expose a compact structured interface (Fig. 3). `Divide`(I, Q) splits the current view into the agent-selected quadrants $Q \subseteq \{$`UL`, `UR`, `LL`, `LR`$\}$, with a small overlap band and a non-overlapping *core* for boundary ownership. `Detect` terminates a branch by emitting local centres and boxes. `Merge` maps all leaf predictions back to the original frame, keeps predictions whose centres fall in their owned core, and applies class-aware NMS, returning the global count $\hat{n}$ and detection set $\hat{\mathcal{D}}$.

Agent action space. At every turn the policy $\pi_\theta(a \mid I, h)$, conditioned on the current view $I^{(d)}$ at recursion depth d and the in-trajectory history h, emits exactly one of two structured calls:

- `Divide`(Q) for some $Q \subseteq \{$`UL`, `UR`, `LL`, `LR`$\}$: the agent will be re-invoked on each `Divide`$(I^{(d)}, q)$ at depth $d{+}1$ in DFS order.
- `Detect`: terminate this branch and contribute the local detections to the global pool.

A maximum recursion depth $D_{\max}$ caps the tree size; once it is reached the agent is forced to choose `Detect`. The single root-level `Merge` is deterministic and not a learnable action.

Quadtree rollout. The sequence of (view, action) pairs traversed in DFS order forms a quadtree T: internal nodes are `Divide` calls, leaves are `Detect` calls, and the root-level `Merge` emits the final answer. The tree depth and branching factor are determined by the policy, so sparse images can terminate immediately while dense regions recurse.

Coordinate bookkeeping. Each leaf prediction is stored with its root-to-leaf quadrant transforms. `Merge` composes these transforms, applies the core-ownership rule, and runs NMS on the surviving predictions. Unlike count summing, this explicitly deduplicates boundary objects visible in multiple sibling crops.

Why *adaptive* recursion? Always dividing wastes calls on empty or easy regions, while never dividing forfeits resolution. The adaptive choice, learned by RL, is also what makes OPD-based preservation possible: on knowledge or reasoning inputs the merged agent can choose `Detect` immediately and behave like the base VLM.

3.2 MULTI-TURN TOOL-USE REINFORCEMENT LEARNING

Trajectory and policy. A trajectory τ is a sequence of (view, action) pairs $\tau = \{(I_k^{(d)}, a_k^{(d)})\}$ traversing the quadtree in DFS order, where each $a_k^{(d)}$ is either a `Divide` or a `Detect` tool call (Fig. 3). The autoregressive policy $\pi_\theta(a \mid I, h)$ conditions on the current image and the history of previous outputs in the same root-to-current path.

Reward design. Given ground-truth boxes $\mathcal{D}^* = \{(\mathbf{b}_i^*, c_i^*)\}_{i=1}^N$ and merged predicted boxes $\hat{\mathcal{D}} = \{(\hat{\mathbf{b}}_j, \hat{c}_j, \hat{s}_j)\}_{j=1}^M$, we define:

$$r_{\text{cnt}}(\tau) = \exp\Big(-\beta_c \, \frac{|N-M|}{\max(N,1)}\Big), \tag{1}$$

$$r_{\text{det}}(\tau) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\big(\hat{c}_{\sigma^*(i)} = c_i^*\big) \cdot \text{IoU}(\hat{\mathbf{b}}_{\sigma^*(i)}, \mathbf{b}_i^*), \tag{2}$$

$$\text{where} \quad \sigma^* = \arg\min_{\sigma \in \mathfrak{S}_M} \sum_i \mathcal{L}_{\text{match}}\big((\mathbf{b}_i^*, c_i^*), (\hat{\mathbf{b}}_{\sigma(i)}, \hat{c}_{\sigma(i)})\big), \tag{3}$$

where $\mathcal{L}_{\text{match}}$ is the standard DETR matching cost ($\lambda_{\text{cls}}\,\mathbb{I}(\cdot) + \lambda_{L1}\,\|\cdot\|_1 + \lambda_{\text{IoU}}\,(1-\text{IoU})$, with the same 1:5:2 ratio as in DETR (Carion et al., 2020)) and σ^* is computed by the Hungarian algorithm. For *counting-only* datasets without box ground truth (LVLM-Count), $r_{\text{det}} \equiv 0$ and we rely on r_{cnt} alone.

The two terms provide complementary pressure on the same rollout: count rewards global cardinality after `Merge`, while detection rewards spatially faithful leaves.

End-to-end final reward. We deliberately keep the reward minimal and end-to-end. The full reward is

$$R(\tau) = \lambda_c \, r_{\text{cnt}}(\tau) + \lambda_d \, r_{\text{det}}(\tau), \tag{4}$$

with $(\lambda_c, \lambda_d) = (0.4, 0.6)$ on detection datasets and $(1.0, 0)$ on counting-only ones. No process-level or step-level shaping is added: the model receives a single scalar reward at the end of the rollout, computed on the global `Merged` set. Adaptive recursion emerges because the reward gives no extra credit for needless depth and the KL term anchors the policy to the RFT cold start.

Multi-turn GRPO. We optimise π_θ via GRPO (Shao et al., 2024), sampling $G{=}8$ rollouts per image and computing the group-normalised advantage

$$\hat{A}_i = \frac{R(\tau_i) - \mu_R}{\sigma_R + \epsilon}, \quad \mu_R = \frac{1}{G}\sum_i R(\tau_i), \tag{5}$$

which is then broadcast uniformly to every learnable action token in τ_i (both `Divide` and `Detect` tokens). The policy loss is the standard clipped objective:

$$\mathcal{J}(\theta) = \mathbb{E}\Big[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|\tau_i|} \sum_{t=1}^{|\tau_i|} \min\big(\rho_{i,t}\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1{-}\epsilon, 1{+}\epsilon)\hat{A}_i\big)\Big] - \beta\,\mathbb{D}_{\text{KL}}. \tag{6}$$

KL is taken against the RFT cold-start reference policy π_{θ_0}. We cache visual encoder outputs across turns within a trajectory to make multi-turn training tractable.

Cold-start SFT via rejection-sampling fine-tuning (RFT). Pure RL from the Qwen3-VL-8B-Thinking base tends to collapse to shallow trajectories. We therefore precede RL with a rejection-sampled cold start: for each training image, we sample candidate `Divide`/`Detect`/`Merge` trajectories, score them with Eq. 4, and fine-tune on the best high-reward trajectory. This primes valid tool use and coarse recursion; GRPO then refines depth and quadrant choices. Implementation details are in Appendix B.

3.3 FOUR-TEACHER OPD MERGING

Setup. After RL we obtain three tool-use specialists $\theta_1, \theta_2, \theta_3$ for SKU-110K, SOD-Aerial, and LVLM-Count, all initialised from the same Qwen3-VL-8B base θ_0. Direct SFT and weight-level

merging retain some dense-perception gain but strongly degrade general capability (Tab. 1). We therefore consolidate the specialists into one student with Online Policy Distillation.

Online Policy Distillation with four teachers. Let the four teachers be $\{T_i\}_{i=0}^{3}$ with $T_0 = \theta_0$ (the base anchor) and $T_{1,2,3} = \theta_{1,2,3}$ (the RL specialists). We initialise the student from the base, $\theta_S \leftarrow \theta_0$, and define a calibration data mixture

$$\mathcal{D}_{\text{calib}} = w_1\,\mathcal{D}_{\text{SKU}} \cup w_2\,\mathcal{D}_{\text{SOD}} \cup w_3\,\mathcal{D}_{\text{LVLM}} \cup w_0\,\mathcal{D}_{\text{base}}, \tag{7}$$

where $\mathcal{D}_{\text{base}}$ aggregates a small held-out subset of MMMU, MMMU-Pro, MMStar, and MMBench-V1.1, and the mixture weights are tuned on validation (default $w_1{=}w_2{=}w_3{=}0.2$, $w_0{=}0.4$). CountBench is excluded from this mixture and used only for held-out transfer evaluation. Each input $x \in \mathcal{D}_{\text{calib}}$ is routed to its owning teacher $T_{\rho(x)}$ by dataset of origin: $\rho(x){=}i$ if $x \in \mathcal{D}_i$ for $i \in \{1,2,3\}$, and $\rho(x){=}0$ for general/knowledge inputs.

At each distillation step we sample a batch from $\mathcal{D}_{\text{calib}}$, roll out the student, query the routed teacher on the same input under the *same* `Divide`/`Detect`/`Merge` API, and minimise the on-policy KL divergence between the two action distributions:

$$\mathcal{L}_{\text{OPD}}(\theta_S) = \mathbb{E}_{x\sim\mathcal{D}_{\text{calib}},\, a\sim\pi_{\theta_S}(\cdot|x,h)}\ \mathrm{KL}\Big(\pi_{T_{\rho(x)}}(\cdot\,|\,x,h)\ \Big\|\ \pi_{\theta_S}(\cdot\,|\,x,h)\Big), \tag{8}$$

where h is the in-trajectory tool-call history. The merged model has the same parameter count as the base because θ_S is a single 8B checkpoint.

Why "four teachers"? The base teacher makes distillation input-conditional rather than purely additive. Without it, the student only imitates specialist tool-use behaviour and forgets knowledge/reasoning inputs. With it, general inputs pull the student toward the base distribution, providing an explicit "answer normally" signal while the specialists install dense-perception behaviour.

Task-conditional tool use at evaluation. At evaluation time, the merged student uses the same API as during training. Its first-turn action is task-dependent: `Divide` on dense/counting inputs and `Detect` on many knowledge-and-reasoning inputs. This is inherited from routed teachers rather than imposed by a hand-crafted gate.

Generalisation through the tool policy. CountBench (Paiss et al., 2023) is not used as a specialist task or OPD calibration source, yet the merged model improves on it (Tab. 1). The mechanism is tool-level rather than dataset-level: a held-out counting prompt triggers the same `Divide`/`Detect`/`Merge` policy learned from dense/counting specialists. This is what weight-level merging lacks: an input-conditioned policy deciding when a capability should fire.

Training pipeline summary. Appendix A gives the full pipeline: RFT cold-start SFT, per-dataset multi-turn GRPO, and four-teacher OPD into a single deployable model.

4 Experiments

We organise our evaluation around three questions. **Q1** (§4.2): Does the merged DCAgent-OPD model improve dense perception while preserving generalist capability? **Q2** (§4.3): How does each design choice in the agent's tool interface affect performance (`Divide` overlap ratio, `Merge` NMS-IoU threshold)? **Q3** (§4.4): How do the per-task training stages compose, and why does the joint count+detection reward train better than either reward alone?

4.1 Experimental Setup

Datasets. We train and evaluate on three dense-perception benchmarks, four test-only mainstream VLM benchmarks, and held-out CountBench.

- **SKU-110K** (Goldman et al., 2019): dense retail product detection, $\sim$147 instances per image, single class. Metric: COCO mAP at IoU 0.5:0.95 ($\times$100).
- **SOD-Aerial** (Zhao et al., 2023): small-object aerial detection, multi-class. Metric: COCO mAP ($\times$100).
- **LVLM-Count** (Ghofrani et al., 2025): aggregates FSC-147, TallyQA, Pascal-VOC counting, and Emoji-Bench. Metric: count MAE.

Table 1: **Main results.** Detection: COCO mAP at IoU 0.5:0.95 ($\times 100$). Counting: MAE (lower is better). CountBench: exact-match accuracy ($\times 100$). The merged DCAGENT-OPD model improves dense perception while keeping mainstream VLM regression small; direct SFT gains less on dense tasks and forgets substantially more. "—" = not available.

Model	Mainstream VLM benchmarks				Dense perception (OPD target)			
	MMMU ↑ (val)	MMMU-Pro ↑	MMBench ↑ (V1.1, EN)	MMStar ↑	SKU ↑ mAP	SOD ↑ mAP	LVLM ↓ MAE	CountBench ↑ Acc
Closed-source proprietary VLMs (zero-shot, no tool API)								
Claude-Sonnet-4.6	—	74.5	—	—	15.01	0.00	21.90	89.9
GPT-5.5	—	83.2	—	—	19.04	0.00	9.72	92.5
Gemini-3.1 Pro	—	80.5	—	—	**34.02**	0.08	8.76	**94.1**
Open-source 8B baselines								
Qwen3-VL-8B (base, the 4th teacher)	**74.1**	60.4	**85.3**	**75.3**	2.40	0.02	29.80	91.5
Qwen3-VL-8B + direct SFT	53.4	31.2	65.4	62.5	15.40	1.10	15.60	80.4
DCAGENT-OPD (4-teacher, ours)	73.1	58.6	81.4	71.8	32.50	**5.20**	**5.15**	93.2

- **General and transfer benchmarks**: **MMMU** (val) (Yue et al., 2024a), **MMMU-Pro** (standard) (Yue et al., 2024b), **MMStar** (Chen et al., 2024), **MMBench-V1.1** (EN dev) (Liu et al., 2024), and held-out **CountBench** (Paiss et al., 2023).

Models. The base policy and fourth "anchor" teacher in OPD is **Qwen3-VL-8B-Thinking** (Bai et al., 2025)—the reasoning-trace variant of Qwen3-VL-8B that natively emits step-by-step thoughts before each structured action, which we found to be a substantially better starting point than the Instruct variant for multi-turn tool-use rollouts. We additionally report frontier proprietary baselines—**Claude-Sonnet-4.6**, **GPT-5.5**, and **Gemini-3.1 Pro**—queried via official APIs at the time of submission with the unified prompts in Appendix C. We also report *Qwen3-VL-8B + direct SFT*, a fine-tuning baseline trained on the same per-image dense annotations as our specialists but without the tool API.

Training. Each specialist is produced by rejection-sampled cold-start SFT followed by multi-turn GRPO in **VeRL** (Sheng et al., 2024). The three specialists are then consolidated by four-teacher OPD using the three dense datasets and a held-out calibration subset of mainstream VLM benchmarks; CountBench is excluded. All runs use a single 32× NVIDIA H200 cluster; full hyperparameters are in Appendix B.

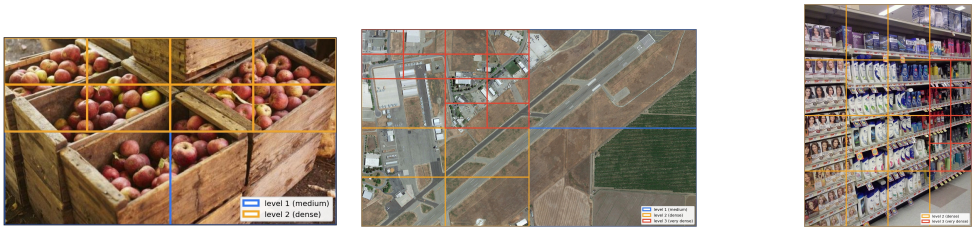
Evaluation protocol. Mainstream VLM benchmarks are evaluated through VLMEvalKit (Duan et al., 2024) for score-template consistency. Dense benchmarks use each dataset's official evaluator. The Qwen3-VL-8B base, all baselines, and the merged DCAGENT-OPD model are queried with the same DCAgent system prompt and tool API exposed (Appendix C); on knowledge-and-reasoning inputs the merged agent typically chooses `Detect` on the first turn and answers in one shot, behaving as an ordinary VLM. Closed-source frontier baselines do not have access to our tool API and are queried zero-shot.

4.2 MAIN RESULTS

Frontier VLMs collapse on dense perception. Table 1 shows a sharp split: proprietary VLMs remain competitive on mainstream and sparse-counting evaluations, but fail on densely packed detection and counting. The open-source base exhibits the same structural weakness.

SFT and RL install the dense-perception tool policy. Before merging, the per-task ablation in Table 1 shows that dense gains are not obtained by direct fine-tuning alone. Direct SFT improves over the zero-shot base, but remains far below the tool-using variants. Introducing the Divide/Detect/Merge interface at the SFT stage already gives a substantial boost, and multi-turn RL further raises the specialist performance to 36.8 mAP on SKU-110K, 7.10 mAP on SOD-Aerial, and 3.10 MAE on LVLM-Count. Thus, SFT mainly provides a valid tool-use cold start, while RL learns the adaptive split-and-stop policy that drives the dense-perception gains.

DCAGENT-OPD improves dense perception without sacrificing the base. After OPD merging, DCAGENT preserves its learned adaptive divide-and-conquer behavior across LVLM-Count, SOD-Aerial, and SKU-110K, selectively applying deeper recursion to locally dense regions instead of uniformly over-splitting the image (Fig. 4). This yields consistent improvements on all three dense-perception targets while keeping mainstream benchmark regression small. By comparison, direct SFT



(a) LVLM-Count (b) SOD-Aerial (c) SKU-110K

Figure 4: **Adaptive divide-and-conquer behaviour after OPD merging.** The OPD-merged DC-AGENT model retains the learned visual-recursion policy across three dense-perception domains. It selectively applies deeper partition levels to locally dense regions while leaving sparse areas coarser. Blue, orange, and red boxes denote increasing recursion depth; levels not visited in a panel are omitted from that panel's legend.



Figure 5: **Qualitative prediction comparison on dense retail scenes.** Compared with open-source and proprietary VLM baselines, DCAGENT-OPD produces denser and better-localised detections in highly crowded SKU-110K scenes.

improves dense-task performance but incurs much larger forgetting, indicating that the tool-policy teacher and base-anchor teacher are both critical for balancing dense specialization with general capability preservation.

Tool use transfers to held-out counting. CountBench (Paiss et al., 2023) is not used as a specialist training target and is excluded from OPD calibration, yet the merged model improves over the base from 91.5 to 93.2. Although this gain is smaller than the improvements on the three dense-perception training domains, it is important because CountBench tests whether the learned behavior transfers beyond the datasets that supervise the specialists. The result suggests that OPD does not merely average task-specific output patterns or memorize dataset formats; instead, it preserves an input-conditional tool policy that can recognize when a new prompt requires instance enumeration and then invoke the learned `Divide`/`Detect`/`Merge` rollout. Trajectories in Appendix C support this interpretation: for counting-style inputs, the merged model activates visual recursion, while ordinary VLM inputs are handled in a single turn. This provides evidence that the consolidated model inherits a reusable dense-perception strategy rather than a collection of dataset-specific specialists.

4.3 TOOL-INTERFACE ABLATIONS

Overlap protects boundaries; `Merge` ownership deduplicates. Table 2 sweeps the overlap ratio between adjacent `Divide` children. No overlap fragments boundary objects; too much overlap spreads each object across more crops and burdens duplicate suppression. A moderate overlap works best with the core-ownership rule.

NMS IoU balances detection and counting. Table 3 sweeps the NMS threshold inside `Merge`. A looser threshold can help detection precision but under-suppresses near-boundary duplicates for counting; the default value gives the best joint trade-off.

Table 2: **`Divide` overlap ratio.** Moderate overlap protects boundary objects, which `Merge` then deduplicates through core ownership.

Overlap ratio	SKU mAP ↑	SOD mAP ↑	LVLM MAE ↓
0%	25.3	4.2	3.1
5%	27.1	5.5	3.4
10% (**ours**)	**36.8**	**7.1**	**3.1**
20%	31.2	6.9	3.9

Table 3: **`Merge` NMS IoU threshold.** The default threshold balances detection precision and duplicate suppression across tasks.

NMS IoU	SKU mAP ↑	LVLM MAE ↓
0.3	33.5	6.4
0.5 (**ours**)	36.8	**3.1**
0.7	**37.1**	5.8

Table 4: **Per-task training-stage ablation.** Per-task scores before OPD show that the tool interface and multi-turn RL drive the dense-perception gains; the merged checkpoint trades a small amount of specialist accuracy for a single deployable model.

Model	**SKU** mAP ↑	**SOD** mAP ↑	**LVLM** MAE ↓
Qwen3-VL-8B (zero-shot)	2.4	0.02	29.8
Qwen3-VL-8B + direct SFT (no tool interface)	12.5	1.10	15.4
DCAGENT w/ SFT only (tool interface, no RL)	20.6	4.50	10.5
DCAGENT w/ RL specialist (tool interface + RL)	**36.8**	**7.10**	**3.10**
DCAGENT-OPD (4-teacher, deployment)	<u>32.5</u>	<u>5.20</u>	<u>5.15</u>

4.4 PER-TASK TRAINING-STAGE ABLATION

Table 4 decomposes the per-task gain. The tool interface already improves over direct SFT, and multi-turn GRPO provides the largest additional step. OPD gives up a small fraction of specialist accuracy to merge three checkpoints into one while preserving general ability (Sec. 4.2).

Count and detection rewards help each other. Figure 2a isolates reward composition under the same initial policy. Count supervision encourages global cardinality, detection supervision encourages spatially grounded leaves, and the joint objective trains cleaner divide-and-merge trajectories than either signal alone.

5 CONCLUSION

We presented DCAGENT, a tool-use VLM agent that learns adaptive divide-and-conquer for dense counting and object detection. Instead of relying on a fixed visual-token budget or a hand-designed crop grid, DCAGENT learns when to `Divide`, when to `Detect`, and how to `Merge` local predictions into a global output. End-to-end count/detection rewards train this multi-turn policy on the final merged result, enabling the model to spend visual resolution where dense objects are most likely to be missed.

Four-teacher OPD further consolidates three dense-perception specialists with a Qwen3-VL-8B base anchor, yielding a single checkpoint that improves dense perception, transfers through tool use to held-out counting, and largely preserves mainstream VLM ability. These results suggest that dense visual perception requires not only stronger visual encoders, but also input-conditional visual computation.

Limitations and broader impact. Our experiments focus on dense counting and object detection, where final outputs can be evaluated with count or box-level rewards. The learned tool policy also increases inference compute when deep recursion is triggered, and may still fail under severe occlusion, domain shift, or extremely small objects. Better dense perception can benefit scientific, retail, aerial, and robotics applications, but deployment in safety-critical or human-sensitive settings should include domain validation, uncertainty awareness, and human oversight.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and migration of the manuscript to the ICLR 2027 LaTeX template. All AI-assisted material was reviewed by the authors, who take responsibility for the final content of this work. **AUTHOR-CONFIRM: disclose any additional use of generative AI in research ideation, literature review, code generation, data processing, experimental design or execution, and result analysis.**

REPRODUCIBILITY STATEMENT

The method, training objectives, and inference procedure are specified in the main paper and Appendix A. Hyperparameters, dataset statistics and splits, baseline protocols, prompts, and additional qualitative examples are provided in Appendices B–D. We will release the implementation, prompts, and trained checkpoints to support reproduction of the reported results.

REFERENCES

Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. Slicing aided hyper inference and fine-tuning for small object detection. In *ICIP*, 2022.

Amin Amini et al. CountGD: Multi-modal open-world counting. In *NeurIPS*, 2024.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

Lin Chen et al. Are we on the right way for evaluating large vision-language models? In *NeurIPS*, 2024.

Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant M Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *AISTATS*, 2019.

Matt Deitke et al. Molmo and PixMo: Open weights and open data for state-of-the-art vision-language models. In *CVPR*, 2025.

Haodong Duan et al. VLMEvalKit: An open-source toolkit for evaluating large multi-modality models. In *ACM MM*, 2024.

Muhammad Ghofrani et al. LVLM-Count: Enhancing the counting ability of large vision-language models. In *Open Review preprint*, 2025.

Eran Goldman, Roei Herzig, Aviv Eisenschtat, Jacob Goldberger, and Tal Hassner. Precise detection in densely packed scenes. In *CVPR*, 2019.

Yang Guo et al. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023.

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. In *ECCV*, 2024.

Xuefeng Li et al. ToRL: Scaling tool-integrated RL. *arXiv preprint arXiv:2503.23383*, 2025.

Yuan Liu et al. MMBench: Is your multi-modal model an all-around player? In *ECCV*, 2024.

Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Arpo: End-to-end policy optimization for gui agents with experience replay. *arXiv preprint arXiv:2505.16282*, 2025.

Michael S Matena and Colin A Raffel. Merging models with Fisher-weighted averaging. In *NeurIPS*, 2022.

Thanh Nguyen, Chau Pham, Khoi Nguyen, and Minh Hoai. Few-shot object counting and detection. In *ECCV*, 2022.

Roni Paiss, Ariel Ephrat, Omer Tov, Shiran Zada, Inbar Mosseri, Michal Irani, and Tali Dekel. Teaching CLIP to count to ten. In *ICCV*, 2023.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *ICLR*, 2024.

Yujia Qin et al. UI-TARS: pioneering automated GUI interaction with native agents. *CoRR*, abs/2501.12326, 2025.

Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *CVPR*, 2021.

Tianhe Ren et al. Grounded SAM: Assembling open-world models for diverse visual tasks. In *arXiv preprint arXiv:2401.14159*, 2024.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *ICLR*, 2016.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Haozhan Shen et al. VLM-R1: A stable and generalizable R1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

Guangming Sheng et al. VeRL: Volcano engine reinforcement learning for LLMs. `https://github.com/volcengine/verl`, 2024.

George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. Model merging by permutation symmetries. In *ICLR*, 2024.

Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging multi-task models via weight-ensembling mixture of experts. In *ICML*, 2024.

F Ozge Ünel, Burak Oguz Ozkalayci, and Cevahir Cigla. The power of tiling for small object detection. In *CVPR Workshops*, 2019.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025a.

Peng Wang et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a.

Weiyun Wang et al. VisualPRM: An effective process reward model for multimodal reasoning. *arXiv preprint arXiv:2503.10291*, 2025b.

Wenbin Wang et al. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. *arXiv preprint arXiv:2408.15556*, 2024b.

Mitchell Wortsman et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal LLMs. *arXiv preprint arXiv:2312.14135*, 2024.

Penghao Wu et al. SEAL: Steerable reasoning calibration of large language models for free. *arXiv preprint arXiv:2410.04039*, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *NeurIPS*, 2023.

B Yang, Z Li, et al. Coda: Coordinating the cerebrum and cerebellum for a dual-brain computer use agent with decoupled reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2026.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. GPT4Tools: Teaching large language model to use tools via self-instruction. In *NeurIPS*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

Le Yu et al. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *ICML*, 2024.

Xiang Yue et al. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*, 2024a.

Xiang Yue et al. MMMU-Pro: A more robust multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2409.02813*, 2024b.

Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, 2016.

Chongqing Zhao et al. SODA: A large-scale benchmark for small object detection. *arXiv preprint arXiv:2207.14096*, 2023.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *ACL System Demonstrations*, 2024.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.

A ALGORITHM: FULL DCAGENT TRAINING AND INFERENCE

We provide the detailed algorithmic specifications for inference (Algorithm 1), per-specialist training (Algorithm 2), and four-teacher OPD consolidation (Algorithm 3).

A.1 INFERENCE: RECURSIVE DIVIDE-AND-CONQUER

Algorithm 1 DCAGENT recursive inference (one image)

Input: image I, policy π_θ, max depth $D_{\max}$, NMS threshold τ_{NMS}, overlap ratio ρ (default 0.10), current depth d, current transform Π, current core C, global pool $\mathcal{P}$
Output: global detection set $\hat{\mathcal{D}}$ (only at root)
Emit decision token $z \in \{\texttt{Divide}, \texttt{Detect}\}$ via $\pi_\theta(\cdot \mid I, d)$
if $z = \texttt{Detect}$ **or** $d = D_{\max}$ **then**
  Sample local predictions $\mathcal{B} = \{(\mathbf{p}_j, \mathbf{b}_j, c_j, s_j)\}$ via the leaf detection prompt (App. C.5)
  for each $(\mathbf{p}_j, \mathbf{b}_j, c_j, s_j) \in \mathcal{B}$ **do**
    $\mathbf{p}_j^{\text{glb}} \leftarrow \Pi \cdot \mathbf{p}_j, \quad \mathbf{b}_j^{\text{glb}} \leftarrow \Pi \cdot \mathbf{b}_j$
    if $\mathbf{p}_j^{\text{glb}} \in C$ **then**
      Add $(\mathbf{p}_j^{\text{glb}}, \mathbf{b}_j^{\text{glb}}, c_j, s_j)$ to $\mathcal{P}$ {Core ownership: only insert if centre lies in current core}
    end if
  end for
else
  Parse $\texttt{Divide}(Q)$ for chosen $Q \subseteq \{\texttt{UL}, \texttt{UR}, \texttt{LL}, \texttt{LR}\}$
  for each $q \in Q$ **do**
    $I_q \leftarrow$ crop of I at quadrant q with overlap band ρ on shared edges
    $\Pi_q \leftarrow \Pi \cdot M_q^\rho, \quad C_q \leftarrow$ non-overlapping core of quadrant q (in global coords)
    INFER$(I_q, \pi_\theta, D_{\max}, \tau_{\text{NMS}}, \rho, d{+}1, \Pi_q, C_q, \mathcal{P})$
  end for
end if
if $d = 0$ **then**
  Root `Merge`: apply class-aware NMS at IoU τ_{NMS} over $\mathcal{P}$ (with the centre-cross rule along $2{\times}2$ split lines)
  return final detections $\hat{\mathcal{D}}$ and count $\hat{n} = |\hat{\mathcal{D}}|$
end if

A.2 PER-SPECIALIST TRAINING: MULTI-TURN GRPO WITH END-TO-END FINAL REWARD

Algorithm 2 DCAGENT multi-turn GRPO training (one specialist)

Input: policy π_θ (cold-started by RFT on the rejection-sampled corpus $\mathcal{D}_{\text{RFT}}$), training set $\mathcal{D}_{\text{tr}}$
Hyperparameters: group size G=8, max depth $D_{\max}$=3, KL coeff β=10^{-3}, lr η=10^{-6}
for step = 1 to $T_{\max}$ **do**
  Sample mini-batch of images $\{I_b\}_{b=1}^{B} \sim \mathcal{D}_{\text{tr}}$
  for each I_b **do**
    // 1. Multi-turn rollout
    for $k = 1, \ldots, G$ **do**
      Sample full quadtree trajectory $\tau_{b,k}$ via Algorithm 1 (sampling, not greedy)
      Aggregate leaf detections $\hat{\mathcal{D}}_{b,k}$ via root-level `Merge`
      // 2. End-to-end final reward
      $r_{\text{cnt}} \leftarrow \exp(-\beta_c\,|N_b - |\hat{\mathcal{D}}_{b,k}||/\max(N_b, 1))$ {Eq. 1}
      $\sigma^* \leftarrow \texttt{Hungarian}(\mathcal{D}_b^*, \hat{\mathcal{D}}_{b,k})$ {Eq. 3}
      $r_{\text{det}} \leftarrow \frac{1}{N_b}\sum_i \mathbb{I}(\hat{c}_{\sigma^*(i)}{=}c_i^*)\,\text{IoU}(\hat{\mathbf{b}}_{\sigma^*(i)}, \mathbf{b}_i^*)$
      $R_{b,k} \leftarrow \lambda_c\, r_{\text{cnt}} + \lambda_d\, r_{\text{det}}$ {No process bonus; Eq. 4}
    end for
    // 3. Group-normalised advantage
    $\mu_R \leftarrow \frac{1}{G}\sum_k R_{b,k}$, $\sigma_R \leftarrow \text{std}(R_{b,\cdot})$
    if $\sigma_R \approx 0$ **then**
      continue {Skip degenerate groups (no learning signal)}
    end if
    $\hat{A}_{b,k} \leftarrow (R_{b,k} - \mu_R)/(\sigma_R + \epsilon)$ for all k
  end for
  // 4. GRPO update
  Update θ with the standard clipped GRPO loss; broadcast $\hat{A}_{b,k}$ to all action tokens of $\tau_{b,k}$.
  Apply KL penalty against the RFT cold-started reference policy.
end for
return θ

A.3 FOUR-TEACHER OPD: ONLINE POLICY DISTILLATION

Algorithm 3 Four-Teacher Online Policy Distillation (OPD)

Input: base θ_0 (the 4th "anchor" teacher), specialists $\{\theta_i\}_{i=1}^{3}$, datasets $\{\mathcal{D}_i\}_{i=1}^{3}$ (SKU/SOD/LVLM-Count), base calibration set $\mathcal{D}_{\text{base}}$ (held-out subset of MMMU/MMMU-Pro/MMStar/MMBench-V1.1; CountBench excluded), mixture weights $\{w_i\}_{i=0}^{3}$, distillation steps T_{OPD}
Output: merged student θ_S
Initialise $\theta_S \leftarrow \theta_0$
Define teacher pool $\{T_0, T_1, T_2, T_3\} = \{\theta_0, \theta_1, \theta_2, \theta_3\}$
Define routing $\rho(x)=i$ if $x \in \mathcal{D}_i$ for $i \in \{1,2,3\}$, $\rho(x)=0$ if $x \in \mathcal{D}_{\text{base}}$
for step $= 1$ to T_{OPD} **do**
  // 1. Sample batch from calibration mixture
  Sample $\{x_b\}_{b=1}^{B} \sim w_0\mathcal{D}_{\text{base}} \cup \bigcup_{i=1}^{3} w_i\mathcal{D}_i$
  for each x_b **do**
    Identify owning teacher: $T_{\rho(x_b)}$
    // 2. On-policy student rollout
    for $k = 1, \ldots, G$ **do**
      Sample student trajectory $\tau_{b,k}^{S} \sim \pi_{\theta_S}(\cdot \mid x_b)$ via Algorithm 1
    end for
    // 3. Teacher action probabilities on the same trajectory prefixes
    For every (view $I_t^{(d)}$, history h_t) along $\tau_{b,k}^{S}$, compute teacher distribution $\pi_{T_{\rho(x_b)}}(\cdot \mid I_t^{(d)}, h_t)$ in a single forward pass
    // 4. Online distillation loss
    $\mathcal{L}_b \leftarrow \frac{1}{G}\sum_k \sum_t \mathrm{KL}\big(\pi_{T_{\rho(x_b)}}(\cdot \mid I_t^{(d)}, h_t) \,\|\, \pi_{\theta_S}(\cdot \mid I_t^{(d)}, h_t)\big)$
  end for
  // 5. Update student
  $\theta_S \leftarrow \theta_S - \eta \nabla_{\theta_S} \frac{1}{B}\sum_b \mathcal{L}_b$
end for
return θ_S

Practical notes. (a) The base teacher θ_0 has no native tool-call experience, so on $\mathcal{D}_{\text{base}}$ it produces single-turn answers; the student is therefore distilled toward "no `Divide` call" on knowledge inputs, which is precisely the desired task-conditional behaviour. (b) For the three specialist teachers, distillation queries the teacher *on the student's own rollout history*—a true on-policy distillation that handles trajectory-level distribution shift. We use G=4 rollouts per input. (c) On Qwen3-VL-8B-Thinking, T_{OPD}=1,000 distillation steps with batch B=32 takes ∼2.5 hours on the same 32× H200 cluster used for the per-specialist RL stage. The merged model has identical parameter count to the base.

B IMPLEMENTATION DETAILS

B.1 DATASET DETAILS AND SPLITS

Split protocol. The unit of splitting is an image, rather than an annotation or a generated tool-use trajectory. All prompts and trajectories derived from the same image therefore remain in the same partition. Specialist SFT and GRPO use only the training partition; development data are used for model selection and hyperparameter tuning; and test data are used only for the final numbers. We preserve each source dataset's released split whenever one is available. The exact image identifiers, preprocessing script, and checksum of every split will be included with the released code.

SKU-110K. We use the corrected `SKU110K_fixed` release. Its 11,743 shelf images contain 1,208,482/90,968/431,546 product boxes in the train/dev/test partitions, respectively (about 147 boxes per image overall). All products share the generic `object` label; "110K" refers to the number of distinct stock-keeping units depicted in the images, not the number of detection classes. We use

the released 8,219/588/2,936 partition directly and report COCO AP averaged over IoU thresholds 0.50:0.05:0.95. No additional random validation split is made.

SODA-A (SOD-Aerial in the main tables). We use the aerial subset of SODA, not SODA-D. It contains 2,513 high-resolution images split by the benchmark into 1,067 training, 576 validation, and 870 test images. The nine evaluated foreground classes are airplane, helicopter, small vehicle, large vehicle, ship, container, storage tank, swimming pool, and windmill; the released `ignore` category is excluded from the reward and evaluation. Because DCAGENT emits horizontal boxes, each annotated quadrilateral is converted to its tight axis-aligned enclosing box before computing COCO AP at IoU 0.50:0.05:0.95. **AUTHOR-CONFIRM: verify that the reported experiments used this horizontal-box conversion, rather than rotated-box AP, and use the same dataset name (preferably SODA-A) throughout the paper.**

LVLM-Count composite. LVLM-Count is a method and evaluation suite rather than a single dataset with a native joint train/dev/test split. Following its evaluation domains, our counting pool draws from FSC-147, TallyQA counting questions, the PASCAL-VOC counting benchmark, and Emoji-Count. FSC-147 itself provides class-disjoint splits of 3,659/1,286/1,190 images; TallyQA provides 249,318 training and 38,589 test questions, potentially with multiple questions referring to the same image; PASCAL-VOC counting contains 102 questions; and Emoji-Count contains 415 examples. We group TallyQA records by source image before sampling so that questions about the same image cannot cross partitions. After filtering and image-level grouping, the current DCAgent protocol contains 4,832/241/1,207 counting examples (Table 6). We evaluate the final integer answer using mean absolute error (MAE).

AUTHOR-CONFIRM—required before submission. The source bundle does not contain the manifest needed to reconstruct the DCAgent-specific LVLM-Count composite. Add, for each component, (i) the retained train/dev/test counts, (ii) the filtering and deduplication rules, (iii) the sampling seed, and (iv) the exact image/question-ID manifest. Confirm that the aggregate 4,832/241/1,207 counts and the image-grouping rule above match the experiments; otherwise replace them with the actual protocol.

Evaluation-only and anchor data. CountBench is used only for held-out transfer evaluation and is excluded from specialist training and OPD calibration. MMMU (validation), MMMU-Pro (standard), MMStar, and MMBench-V1.1 (English development) are evaluated with the splits named in Section 4. The 256 examples per general benchmark used by the base anchor teacher must be disjoint from every example on which a final score is reported. **AUTHOR-CONFIRM: replace `[AUTHOR_PROTOCOL]` with the source split for these anchor examples and release their IDs; if the same public split was used for both calibration and reporting, recompute the score on the untouched remainder and state its size.**

B.2 BASELINE REPRODUCTION NOTES

Closed-source. Claude-Sonnet-4.6, GPT-5.5, and Gemini-3.1 Pro were queried via official APIs with temperature 0 and the unified DCAgent role prompt (Appendix C.1) plus the corresponding dataset-specific task prompt (Appendix C.6). The three visual tools (`Divide`/`Detect`/`Merge`) were not exposed to the proprietary baselines, so they always answer in a single turn. Each image was sent at its native resolution; no client-side downsampling.

Tool-augmented. For LVLM-Count we used the authors' code with the recommended Grounding-DINO + SAM weights. For DC^2 we re-implemented the patch-describe-filter pipeline at 4×4 grid (the best HR-Bench setting). Both ran on top of Qwen3-VL-8B-Thinking, identical to our base.

VLMEvalKit. For the four mainstream VLM benchmarks and CountBench (Tab. 1), all entries (base, direct-SFT baseline, our merged checkpoint, and the per-stage variants in Tab. 4) were evaluated through the same VLMEvalKit (Duan et al., 2024) commit `v0.4.5` with default judges (GPT-4o for MMVet-style metrics; rule-based for MMMU/MMStar/MMBench/CountBench). This guarantees that any score difference reflects actual capability change, not template noise.

B.3 Compute Budget and Carbon Estimate

Total wall-clock for the main experiments was ∼2,450 GPU-hours on H200 (3× specialist training ∼67.5h × 32 GPUs + OPD ∼2.5h × 32 GPUs + evaluation ∼10h × 32 GPUs). We did not observe meaningful gains from longer training within our hyperparameter budget. Full carbon-equivalent estimate: ∼1,150 kg CO_2-eq (using a standard 0.47 kg CO_2/kWh grid average and a 1,000W per-H200 power draw).

C Prompt Templates

This section provides the exact prompts used by DCAgent. The goal is reproducibility: the model is not given a fixed crop schedule, but instead decides when to call visual tools, which subviews to inspect, and when to return the final merged answer.

C.1 Agent Role Prompt

At the beginning of each episode, the model receives a task instruction and the current image view. The system prompt defines the model as a visual divide-and-conquer agent.

Listing 1: Agent role / system prompt

```
You are DCAgent, a vision-language agent for dense visual
perception.
Your goal is to count and/or localise every target object in the
image.
Dense scenes may contain many small, tightly packed, or partially
occluded objects.  You may use visual tools to improve recall:
- Divide:  split the current visual view into four subviews and
recursively solve each subview.
- Merge:  combine the results from subviews into one global
prediction, removing duplicate objects that appear near crop
boundaries.
Use tools only when they are helpful.  If the current view is
sparse or already clear, solve it directly.  If the current view is
crowded, contains many tiny objects, or objects are hard to separate
at the current resolution, call Divide.
Always preserve global object identity:  each physical object should
appear once in the final answer.
```

The user-side task prompt specifies the dataset target category and output format. The policy can either emit a direct answer or a tool call. During training, tool calls are executed by the environment and the resulting subview observations are appended to the dialogue.

C.2 Tool 1: Divide

`Divide` is the visual exploration tool, exposed to the model as a function-style tool:

Listing 2: `Divide` tool schema

```
{
"name": "Divide",
"description": "Split the current image view into four 2x2 subviews
for recursive dense perception.",
"arguments": {
"view_id": "the current image view identifier",
"reason": "why the current view should be divided"
},
"returns": {
"children": [
{"view_id": "top_left", "position": "top-left subview"},
{"view_id": "top_right", "position": "top-right subview"},
{"view_id": "bottom_left", "position": "bottom-left subview"},
{"view_id": "bottom_right", "position": "bottom-right subview"}
]
}
}
```

Operationally, the environment divides the current view into a 2×2 grid. Each child crop includes a small overlap with its neighbouring crops (default 10% of the parent linear extent on each shared edge) so that objects near the split boundary remain visible. Each child also has a non-overlapping ownership region, called its *core*. The core is used by `Merge` to decide which crop owns an object whose centre is near a boundary.

The model is encouraged to call `Divide` when:

- the estimated number of target objects is high;
- target objects are small relative to the full image;
- adjacent objects are visually merged at the current resolution;
- boundary or occlusion cases make direct enumeration unreliable.

The model is encouraged to avoid `Divide` when:

- the scene is sparse;
- the task is primarily knowledge, OCR, or reasoning rather than dense perception;
- the current view is already a small leaf crop;
- further splitting would cut most target objects into ambiguous fragments.

C.3 Tool 2: Merge

`Merge` aggregates leaf predictions into the coordinate frame of the original image:

Listing 3: `Merge` tool schema

```
{
"name": "Merge",
"description": "Merge predictions from four child views into the
parent view and remove duplicate boundary detections.",
"arguments": {
"parent_view_id": "the parent image view",
"child_results": [
{
"view_id": "child view identifier",
"detections": [
{
"label": "target category",
"point": "[x, y] in the current coordinate frame",
"bbox": "[x1, y1, x2, y2] in the current coordinate frame",
"confidence": 1.0
}
]
}
]
},
"returns": {
"detections": "merged class-aware detections in the parent/global
coordinate frame",
"count": "number of merged detections"
}
}
```

The merge rule has three parts:

1. **Coordinate restoration.** Leaf predictions are first mapped from local crop coordinates back to the parent (or original) image coordinate frame using the recorded sequence of 2×2 scaling+translation transforms.

2. **Core ownership.** A prediction is counted by the child crop whose core contains the object centre. Predictions visible only in the overlap band are discarded unless their centre belongs to that crop's core.

3. **Duplicate suppression.** Remaining predictions are merged using class-aware non-maximum suppression at IoU 0.5. For point-only outputs, nearby points of the same class are clustered. A wider centre-cross rule is used along the vertical and horizontal 2×2 split lines, because the same object is often clicked once on each side of the boundary.

The final count is always the number of merged objects, not the raw sum of the four child counts.

C.4 Decision Prompt

Before solving a view, DCAgent receives a decision prompt that asks whether the view should be divided or solved directly:

Listing 4: Per-turn decision prompt

```
You are controlling a divide-and-conquer visual grounding procedure
for dense perception.
Available tool:
1. Divide: split the current crop into four overlapping 2x2
sub-crops, recursively solve each sub-crop, then Merge will combine
the four results and remove center-boundary duplicates.
Current crop metadata:
- target category: {category}
- recursion depth: {depth}/{max_depth}
- crop size: {width}x{height} pixels
- recommended divide threshold: more than about {threshold} visible
instances, very small objects, heavy occlusion, or crowded layout.
Decide whether this crop should call Divide again or be solved
directly now. Use Divide only when another 2x2 split is likely
to improve recall or localisation.
Return strict JSON only:
{
"tool": "Divide" or "Detect",
"estimated_count": <integer or null>,
"reason": "<short reason>"
}
```

This prompt is used only to expose the two possible actions. The recursive policy is learned: during GRPO rollouts, the model receives reward only from the final merged output, so unnecessary division, missed objects, and duplicate boundary counts are all penalised through the end task reward.

C.5 Leaf Detection Prompt

When the model decides to solve a crop directly, it receives a leaf-level grounding prompt:

Listing 5: Leaf-level detection prompt

```
This image may be a crop from a larger image. Treat this crop as
the current image.
Locate every distinct {category}. For each object, emit exactly
one centre point using <point>x y</point> with integer coordinates
normalised to [0, 1000]. If bounding boxes are requested, also
provide [x1, y1, x2, y2] in the same normalised coordinate system.
Include visible objects whose centres are inside the crop, including
partially visible objects. Boundary duplicates will be removed
later by Merge.
Do not output anything except the numbered object list.
```

For detection datasets, the point is used as the object centre and the optional box is used for localisation scoring. If only points are emitted, a small category-agnostic box centred at the point is used for detection evaluation. For counting datasets, the final count is the number of merged points.

C.6 Dataset-Specific Task Prompts

SKU-110K

```
This is a retail shelf image with densely packed products.
Count and localise every individual product on the shelf. Treat
each separate physical item as one object, even when products are
adjacent, partially occluded, or visually similar.
```

SOD-Aerial

```
This is a high-resolution aerial image.
Count and localise every object of interest, including vehicles,
ships, aircraft, persons, buildings, storage tanks, swimming
pools, windmills, and other annotated small aerial objects.  Ignore
background clutter and unlabelled regions.
```

LVLM-Count

```
Count every visible {object_of_interest} in the image.
Emit one centre point for each instance.  The total number of merged
points is the final count.
```

C.7 Final Answer Format

DCAgent's final response after all tool calls is a compact JSON object:

```
{
"detections":  [
{
"label":  "object",
"point":  "[x, y]",
"bbox":  "[x1, y1, x2, y2]"
}
],
"count":  0
}
```

Coordinates are normalised to the original image coordinate frame unless otherwise stated. The `count` field must equal the number of merged detections.

C.8 Training Reward Interface

Intermediate tool choices are not scored independently. The reward is computed on the final merged output:

- **Counting reward:** negative count error, derived from mean absolute error between predicted and ground-truth counts.
- **Localisation reward:** DETR-style Hungarian matching between predicted and ground-truth objects, using box IoU after `Merge`.
- **Duplicate penalty:** duplicate predictions that match the same ground-truth object reduce both count and localisation reward.
- **Miss penalty:** ground-truth objects without a matched prediction reduce recall and increase count error.

This makes `Divide` useful only when it improves the final merged prediction. The agent therefore learns to trigger tools on dense visual inputs and to bypass them on ordinary visual question answering or knowledge-and-reasoning inputs.

D Additional Qualitative Results

This section provides additional qualitative examples on the three dense-perception benchmarks used in our evaluation: LVLM-Count, SOD-Aerial, and SKU-110K. These examples complement the quantitative results in the main paper by visualizing model predictions on individual dense scenes. For each example, the title above each panel reports the model prediction and the corresponding error or score when available.

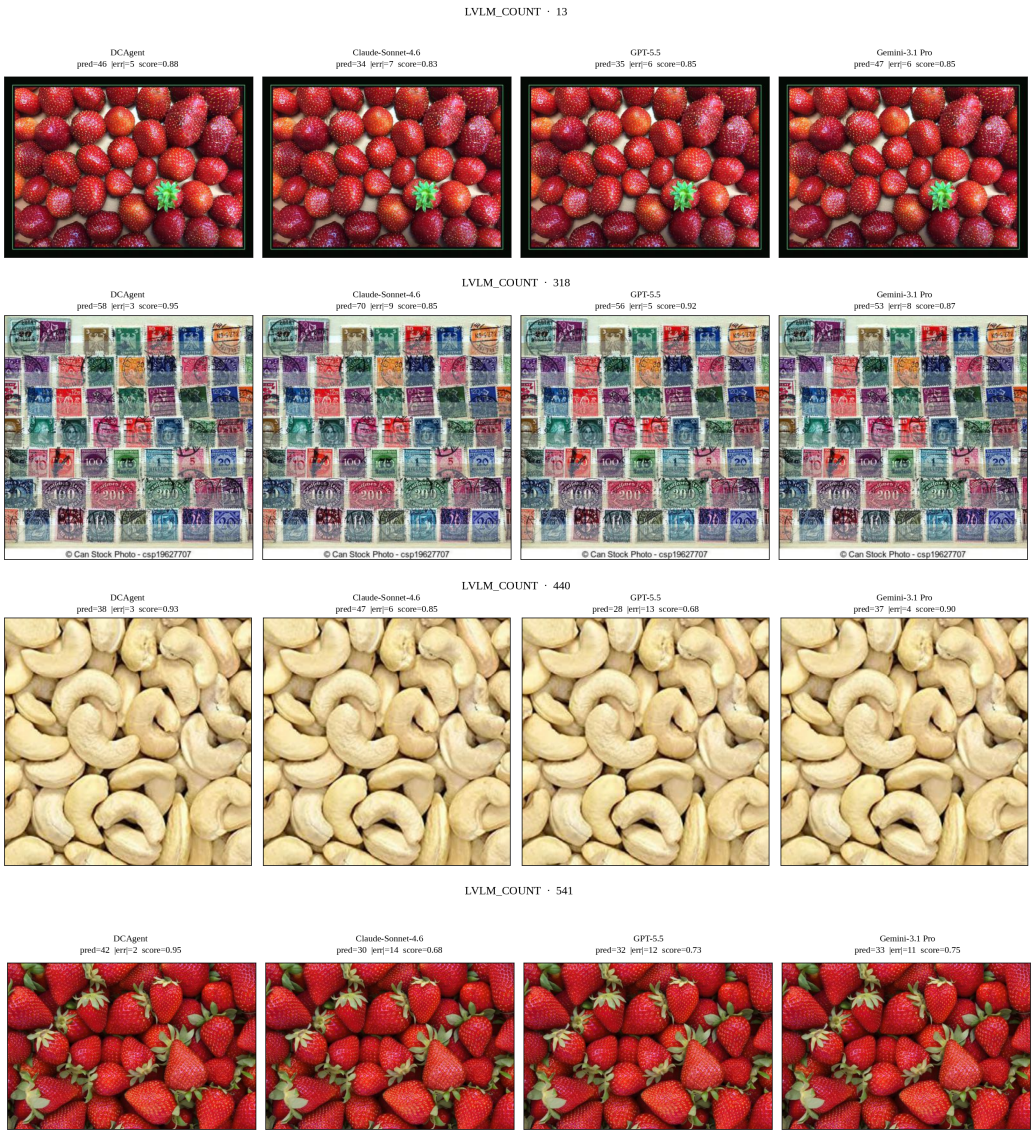


Figure 6: Additional qualitative counting results on LVLM-Count. Each row shows one example. These examples contain repeated objects, strong occlusion, and ambiguous instance boundaries, making precise counting challenging from a single global view.

These qualitative examples are intended to make the failure modes of dense visual perception more interpretable. In dense scenes, models may produce visually plausible but quantitatively inaccurate outputs, such as under-counting small objects, missing crowded instances, or duplicating objects near crop boundaries.

D.1 Additional Qualitative Results on LVLM-Count

LVLM-Count focuses on dense object counting. The examples below contain many repeated objects with occlusion, local similarity, and ambiguous boundaries. These cases highlight the difficulty of precise instance enumeration from a single global view.



Figure 7: Additional qualitative detection results on SOD-Aerial, part 1. Each row shows one aerial scene. These examples illustrate the difficulty of detecting small objects embedded in large high-resolution views.

D.2 Additional Qualitative Results on SOD-Aerial

SOD-Aerial emphasizes small-object detection in aerial imagery. Objects are often tiny relative to the full image and can be easily lost under a fixed visual-token budget. The examples below illustrate the challenge of localizing small targets in large, cluttered scenes.

D.3 Additional Qualitative Results on SKU-110K

SKU-110K contains dense retail shelf images with many visually similar product instances. These examples test whether a model can localize many small objects under strong repetition, occlusion, and boundary ambiguity. They also expose common dense-detection errors, including missed instances and duplicated predictions.

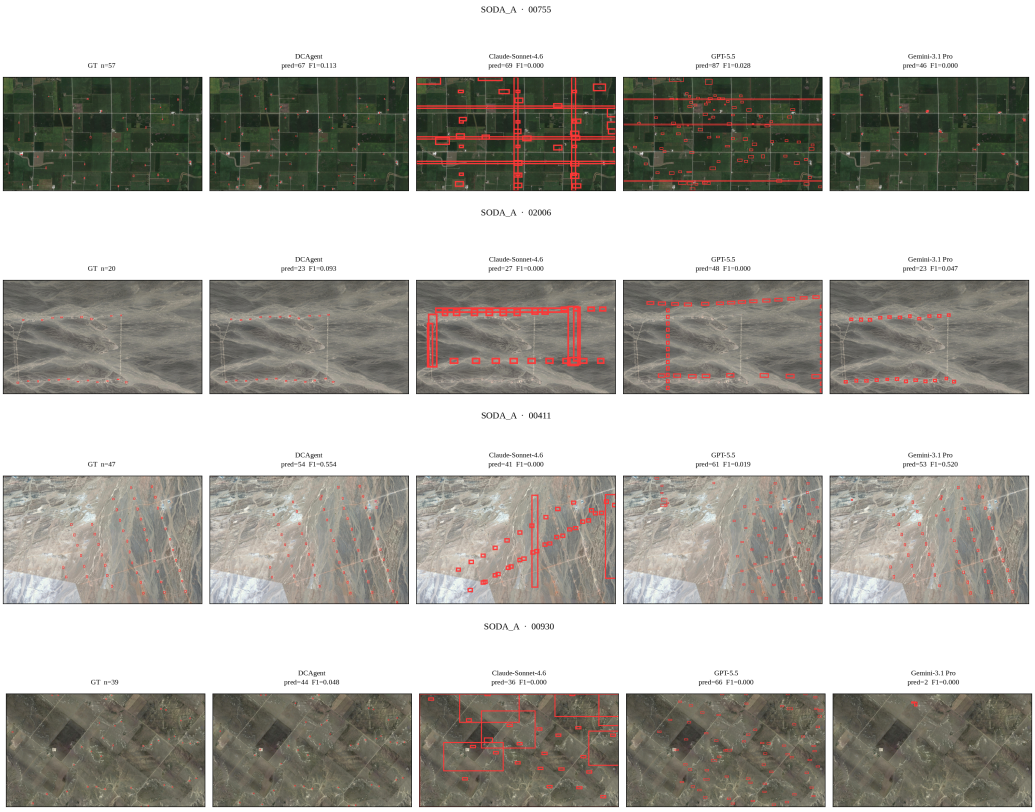


Figure 8: Additional qualitative detection results on SOD-Aerial, part 2. Dense aerial scenes require allocating visual resolution to small target-rich regions rather than processing the full image uniformly.

Table 5: Detailed hyperparameters for DCAGENT training.

Hyperparameter	**Value**
Cold-Start SFT via Rejection-Sampling Fine-Tuning (RFT)	
Backbone	Qwen3-VL-8B-Thinking
Framework	LlamaFactory (Zheng et al., 2024) (full-parameter, packed sequences)
Rollouts per training image G_{rft}	16
Acceptance criterion	top-1 trajectory per image with composite reward $R \geq 0.6$ (Eq. 4)
Retained trajectories per dataset	3,200 (SKU-110K) + 2,800 (SOD-Aerial) + 1,200 (LVLM-Count) = 7,200 total
Optimiser	AdamW
Learning rate	1×10^{-5}, cosine decay
Batch size	32
Epochs	1
Sampling temperature (rollouts)	1.0
Multi-Turn Tool-Use GRPO (per specialist, end-to-end RL)	
Policy backbone	Qwen3-VL-8B-Thinking (RFT cold-started)
Framework	VeRL (Sheng et al., 2024) (modified for multi-turn rollouts)
Group size G	8
Mini-batch (images)	32 (256 trajectories)
Total steps	400 (per specialist)
Optimiser / lr	AdamW / 1×10^{-6}
LR scheduler	Cosine to 1×10^{-7}
KL coefficient β	1×10^{-3}
GRPO clip ϵ	0.2
Max recursion depth $D_{\max}$	3
`Divide` overlap ratio ρ	0.10 (10% of parent linear extent on each shared edge)
`Merge` NMS IoU threshold τ_{NMS}	0.5
Centre-cross rule along split lines	enabled (point-distance threshold 0.02 in normalised coords)
Reward weights (λ_c, λ_d)	detection datasets: $(0.4, 0.6)$; counting-only: $(1.0, 0)$
Hungarian matching cost coeffs $(\lambda_{\text{cls}}, \lambda_{L1}, \lambda_{\text{IoU}})$	$(1, 5, 2)$ (DETR default)
β_c (in r_{cnt})	2.0
Coordinate normalisation	integer in $[0, 1000]$ for emitted points / boxes
Four-Teacher OPD *(Online Policy Distillation)*	
Calibration mixture $\mathcal{D}_{\text{calib}}$	$w_1\mathcal{D}_{\text{SKU}} + w_2\mathcal{D}_{\text{SOD}} + w_3\mathcal{D}_{\text{LVLM}} + w_0\mathcal{D}_{\text{base}}$
Mixture weights (w_0, w_1, w_2, w_3)	$(0.4, 0.2, 0.2, 0.2)$ default
$\mathcal{D}_{\text{base}}$ source	256 held-out samples / benchmark from MMMU, MMMU-Pro, MMStar, MMBench-V1.1; CountBench excluded
Routing $\rho(\cdot)$	Dataset-of-origin: $\rho(x)=i$ if $x \in \mathcal{D}_i$ for specialists, $\rho(x)=0$ for base inputs
Distillation steps T_{OPD}	1,000
Batch size B	32
Rollouts per input G_S	4
Distillation objective	On-policy KL: $\text{KL}(\pi_{T_{\rho(x)}} \,\|\, \pi_{\theta_S})$ at every action token
Optimiser / lr	AdamW / 5×10^{-7}
LR scheduler	Cosine to 5×10^{-8}
Sampling temperature (teacher & student)	0.7 during training, 0.0 at evaluation
Compute	
Hardware	32× NVIDIA H200 (141 GB) – single cluster used for all stages
RFT cold-start wall-clock	∼1.5h per specialist (rollouts ∼1h + LlamaFactory SFT ∼0.5h)
GRPO RL wall-clock	∼21h per specialist (400 steps, G=8 rollouts/image)
OPD distillation wall-clock	∼2.5h (T_{OPD}=1,000 steps, B=32, G_S=4

Table 6: Dense-perception data used for specialist training and evaluation. Counts are images for SKU-110K and SODA-A and image-level counting examples for the LVLM-Count composite. The asterisk marks a DCAgent-specific composite rather than a native split released by the source datasets.

Dataset	Task and annotation	Train	Dev	Test
SKU-110K (Goldman et al., 2019)	Single-class retail-product detection; horizontal boxes	8,219	588	2,936
SODA-A / SOD-Aerial (Zhao et al., 2023)	Nine-class aerial small-object detection; quadrilateral annotations	1,067	576	870
LVLM-Count composite (Ghofrani et al., 2025)	Open-ended object counting; integer count target	4,832*	241*	1,207*

Table 7: Mean $\pm$ standard deviation across three seeds for the dense-perception main results. Variances are small ($\leq$1.0 absolute on every metric), confirming that the gains in Tab. 1 are not seed-dependent.

Method	SKU mAP	SOD mAP	LVLM MAE
Qwen3-VL-8B (base)	2.40$\pm$0.20	0.02$\pm$0.01	29.80$\pm$0.7
Qwen3-VL-8B + direct SFT	15.40$\pm$0.6	1.10$\pm$0.20	15.60$\pm$0.5
DCAgent w/ SFT only	20.60$\pm$0.5	4.50$\pm$0.30	10.50$\pm$0.6
DCAgent w/ RL specialist	36.80$\pm$0.7	7.10$\pm$0.40	3.10$\pm$0.30
DCAgent-OPD (4-teacher, deployment)	32.50$\pm$0.6	5.20$\pm$0.30	5.15$\pm$0.40

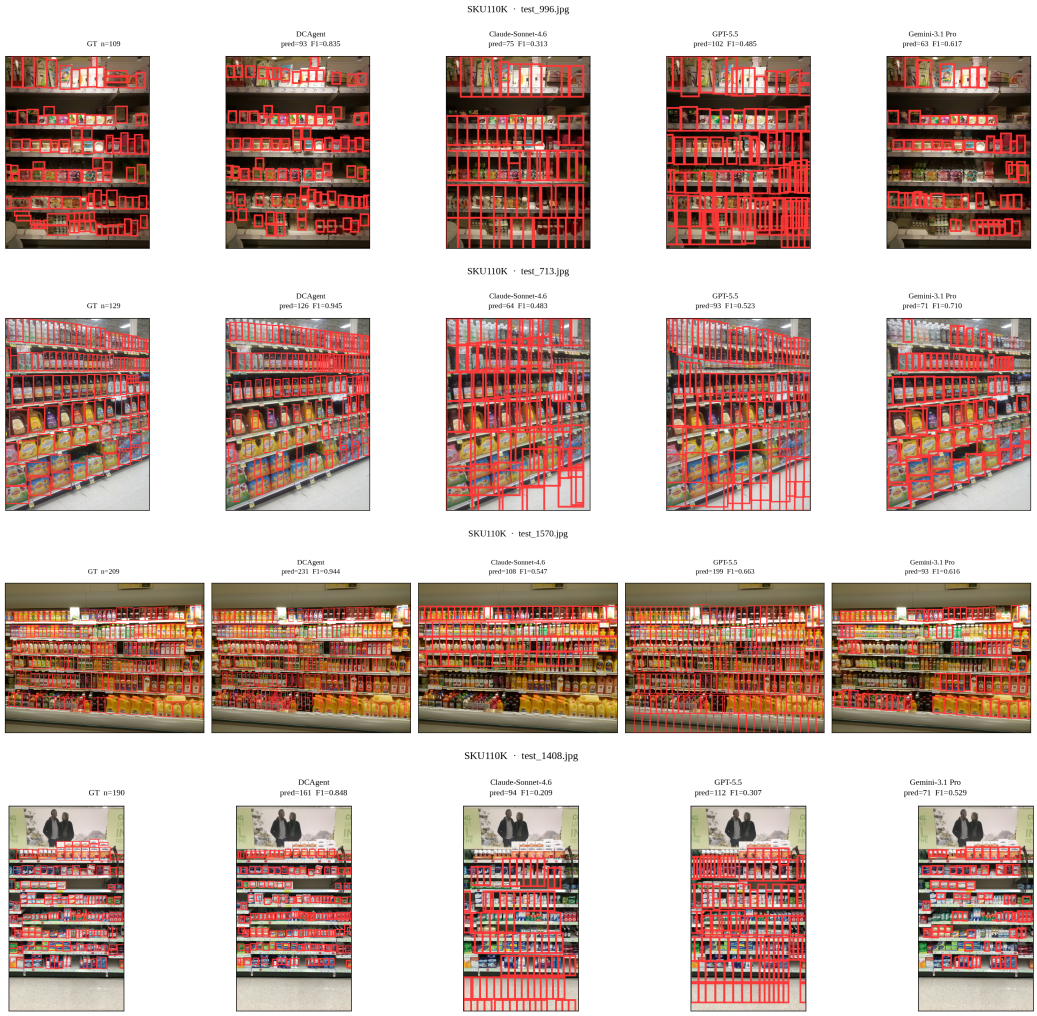


Figure 9: Additional qualitative detection results on SKU-110K, part 1. Each row shows one dense retail shelf example. These scenes contain repeated product patterns, heavy occlusion, and many small object instances.

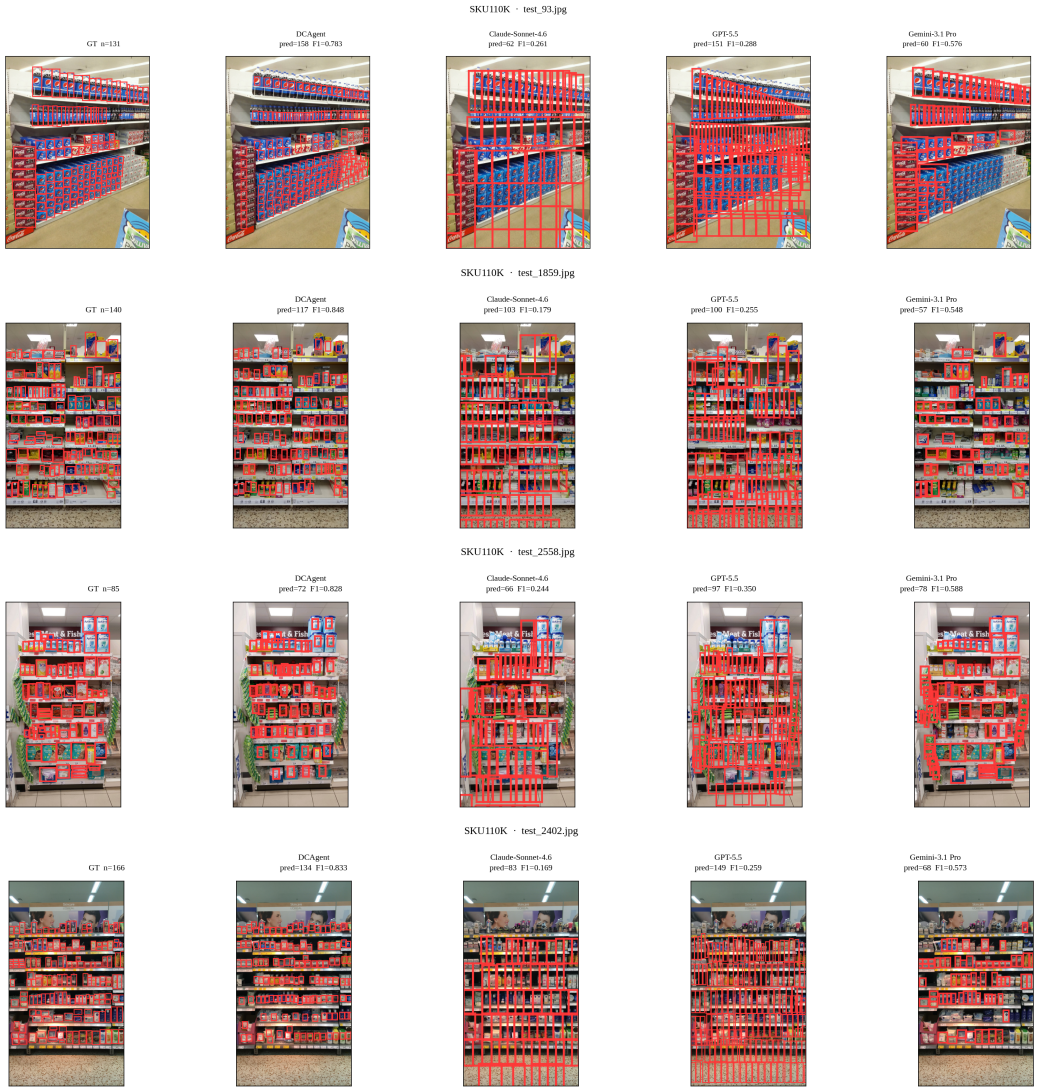


Figure 10: Additional qualitative detection results on SKU-110K, part 2. Dense retail shelves expose common VLM failure modes, including missed instances, duplicated boundary predictions, and confusion among visually similar products.