DetPO: In-Context Learning with Multi-Modal LLMs for Few-Shot Object Detection

Gautam Rajendrakumar Gare[1,⋆], Neehar Peri[1,⋆], Matvei Popov[2], Shruti Jain[1], John Galeotti[1], and Deva Ramanan[1]

[1] Carnegie Mellon University
[2] Roboflow

Abstract. Multi-Modal LLMs (MLLMs) demonstrate strong visual grounding capabilities on popular object detection benchmarks like OdinW-13 and RefCOCO. However, state-of-the-art models still struggle to generalize to out-of-distribution classes, tasks and imaging modalities not typically found in their pre-training. While in-context prompting is a common strategy to improve performance across diverse tasks, we find that it often yields lower detection accuracy than prompting with class names alone. This suggests that current MLLMs cannot yet effectively leverage few-shot visual examples and rich textual descriptions for object detection. Since frontier MLLMs are typically only accessible via APIs, and state-of-the-art open-weights models are prohibitively expensive to fine-tune on consumer-grade hardware, we instead explore black-box prompt optimization for few-shot object detection. To this end, we propose Detection Prompt Optimization (DetPO), a gradient-free test-time optimization approach that refines text-only prompts by maximizing detection accuracy on few-shot visual training examples while calibrating prediction confidence. Our proposed approach yields consistent improvements across generalist MLLMs on Roboflow20-VL and LVIS, outperforming prior black-box approaches by up to 9.7%. Our code is available on GitHub.

Keywords: Few-Shot Object Detection · In-Context Learning · Prompt Optimization · Vision-Language Models

1 Introduction

Multi-Modal LLMs (MLLMs) achieve remarkable zero-shot performance across diverse vision-language tasks like image captioning, OCR, and VQA [1, 5, 11]. On flagship vision tasks like object detection, generalist MLLMs like Qwen3-VL [4] perform on-par with specialist object detectors like GroundingDINO [34] on popular benchmarks like RefCOCO [67] and OdinW-13 [30]. However, such models struggle to generalize to out-of-distribution concepts (e.g. material property

⋆ Equal Contribution

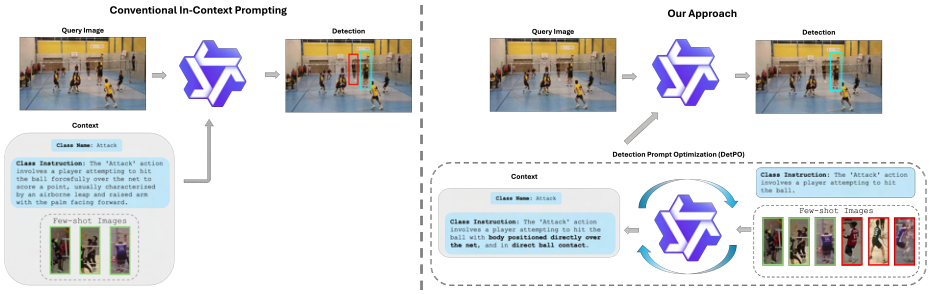


Fig. 1: Detection Prompt Optimization. We cast the problem of gradient-free few-shot object detection as multimodal in-context learning (ICL). Here, a frozen multi-modal LLM (MLLM) is presented with a class name, a textual description, and a few visual examples (**left**), similar to the instructions given to a human annotator tasked with annotating that class [45]. Rather than presenting the visual examples directly to the MLLM, we find that it is far more effective to use them to optimize a better prompt (**right**) via prompt optimization; we use another black-box MLLM to discover prompt instructions that perform better on the few-shot training dataset. These improved instructions are then fed into the target MLLM.

estimation, defect detection, and contextual action recognition) and imaging modalities (e.g. X-rays, thermal spectrum data, and aerial imagery) not typically found in internet-scale pre-training [45]. We argue that some data will *always* remain out-of-distribution, whether due to being sequestered from the internet or being created after a model's training cutoff. This motivates the need to learn from few-shot examples, similar to how human annotators are taught new concepts via few-shot multi-modal instructions [38].

Few-Shot Detection via In-Context Learning (ICL). Since frontier MLLMs are typically only accessible via APIs, and state-of-the-art open-weights models are prohibitively expensive to fine-tune on consumer-grade hardware, we argue that gradient-free few-shot learning is best framed through the lens of multi-modal in-context learning (ICL). Although ICL has been extensively studied for LLMs [8, 14], multi-modal ICL for object detection poses unique challenges. We primarily explore this problem using the recently released Roboflow20-VL (RF20-VL) [45] benchmark, which aggregates 20 distinct datasets from domains typically not found in internet-scale pre-training. Importantly, object classes are specified via few-shot visual examples and rich textual instructions. In practice, we find that prompting with few-shot visual examples yields inconsistent benefits compared to using class names and instructions (Table 1). We posit that this arises from rigid prompt structures used during post-training, making it difficult to exploit additional multi-modal contextual information during zero-shot inference.

Detection Prompt Optimization Improves Concept Alignment. To address these limitations, we introduce Detection Prompt Optimization (DetPO), a black-box optimization approach that iteratively refines text-only prompts by maximizing detection accuracy on few-shot visual training examples. Unlike

Table 1: Multi-Modal ICL Hurts Detection Accuracy. Current MLLMs struggle to consistently benefit from multi-modal in-context examples for object detection. We posit that rigid post-training prompt structures make it difficult to effectively leverage additional context. To address this limitation, we propose Detection Prompt Optimization (DetPO) to encode few-shot multi-modal examples into a single text-only prompt, significantly improving detection accuracy.

Method	Class Names	Instructions	Images	mAP
Qwen2.5-VL [5] (7B)	✓	✗	✗	4.6
Qwen2.5-VL [5] (7B)	✓	✓	✗	6.2
Qwen2.5-VL [5] (7B)	✓	✓	✓	1.8
Qwen2.5-VL [5] (72B)	✓	✗	✗	7.1
Qwen2.5-VL [5] (72B)	✓	✓	✗	10.4
Qwen2.5-VL [5] (72B)	✓	✓	✓	10.1
Qwen3-VL [4] (8B)	✓	✗	✗	10.4
Qwen3-VL [4] (8B)	✓	✓	✗	11.4
Qwen3-VL [4] (8B)	✓	✓	✓	7.0
Qwen3-VL [4] (30B-A3B)	✓	✗	✗	10.7
Qwen3-VL [4] (30B-A3B)	✓	✓	✗	11.9
Qwen3-VL [4] (30B-A3B)	✓	✓	✓	9.8
Gemini 3 Pro [13]	✓	✗	✗	21.9
Gemini 3 Pro [13]	✓	✓	✗	23.0
Gemini 3 Pro [13]	✓	✓	✓	23.9

classical detectors that primarily learn from true positive examples, we find that MLLMs benefit from seeing *corner cases* and *negative examples*, akin to how human annotators refine their understanding of a target class by learning what *not* to annotate. At each iteration, DetPO refines a text-only prompt based on the model's true positive, false positive, and false negative predictions on the few-shot training set.

DetPO Down-Weights False Positive Detections. We further observe that current MLLMs often overpredict bounding boxes and lack per-box confidence scores by default. We find that simply prompting models for per-box scores improves detection accuracy without additional compute cost. However, these self-reported confidence scores can sometimes be poorly calibrated. In such cases, we can optionally post-process detections with VQA Score [33]. We prompt the model with bounding box predictions overlaid on the test image and ask "Is this bounding box an instance of class `{CLS}`?" We use the normalized probability of the `yes` token as the final bounding box confidence score. This simple post-processing step down-weights false positives and further improves mAP.

Contributions. We present three major contributions. First, we benchmark state-of-the-art MLLMs on RF20-VL and LVIS and show that naively prompting with multi-modal in-context examples yields poor performance. Next, we propose Detection Prompt Optimization (DetPO) to iteratively use model predictions on the training set to refine text-only prompts and estimate better per-box confidence scores to improve concept alignment. Lastly, we extensively ablate our

design choices and demonstrate that DetPO consistently improves performance across popular MLLMs, establishing a new state-of-art for black-box few-shot object detection.

2 Related Works

Vision-Language Models are often pre-trained on large-scale, weakly supervised image-text pairs [53] before fine-tuning on task-specific data. While early VLMs mainly focused on image classification [47] and VQA, recent methods have extended these models for visual grounding through open-vocabulary detection. Early efforts adapted VLMs for detection by classifying region proposals [19, 20] or by integrating detection heads into either frozen [26] or fine-tuned [15, 39, 40] encoders. RegionCLIP [70] introduced a multi-stage pipeline that combined pseudo-label generation, region-text contrastive pre-training, and fine-tuning on detection benchmarks. GLIP [31] reformulates detection as a phrase grounding task, where a single text query is applied to the entire image. Detic [71] improves long-tail detection performance by leveraging image-level supervision from ImageNet's 21K classes [50]. Modern VLMs demonstrate strong zero-shot performance and are often employed as "black-box" tools in downstream tasks [24, 37, 42, 43, 56]. More recently, multi-modal large language models (MLLMs) such as Qwen2.5-VL [5], Qwen3-VL [4], Gemini 2.5 Pro [12], and Gemini 3 Pro [13] reframe spatial understanding as a next-token prediction task. Notably, generalist MLLMs demonstrate strong zero-shot visual grounding capabilities on-par with specialist models like GroundingDINO [35] on popular object detection benchmarks like OdinW-13 [30] and RefCOCO [67]. However, we find that these generalist models achieve poor zero-shot detection accuracy on out-of-distribution classes, tasks, and modalities, motivating the need for few-shot concept alignment.

Few-Shot Object Detection (FSOD) focuses on recognizing novel object categories with limited training examples [25]. Prior work primarily explores two paradigms: meta-learning and transfer learning. Meta-learning methods aim to learn transferable representations from `base` classes that can generalize to unseen `novel` classes. For instance, Kang et al. [22] re-weights features from `base` classes to infer `novel` ones, while Xiao et al. [63] unifies few-shot detection and viewpoint estimation. Fan et al. [16] develops a matching-based few-shot object detection network that learns a similarity metric between image pairs, and Wu et al. [62] enhances object features using universal prototypes. Xu et al. [64] further propose a generative framework that improves robustness to noisy object proposals. In contrast, transfer learning methods partially freeze weights pretrained on `base` datasets to enable adaptation to `novel` classes with limited samples. These methods typically adopt a two-stage fine-tuning process, training on `base` classes followed by refinement of the box classifier and regressor using K-shot examples. This strategy has generally outperformed meta-learning approaches [59]. Different from prior work, we explore few-shot object detection for generalist MLLMs using multi-modal in-context learning.

In-Context Learning (ICL) is an emergent capability that enables LLMs to reason by analogy [14]. Brown et al. [8] first popularized few-shot learning via ICL. Subsequent methods extend this idea by improving reasoning through structured prompting, such as decomposing complex instructions into simpler steps [61, 66]. Recently, ICL has also gained traction in the vision-language community. Flamingo [3] demonstrated that large-scale VLMs can perform ICL effectively, improving tasks like image captioning with only a few examples. More recently, Emu 2 [55] demonstrated that scaling encoder-decoder architectures with auto-regressive training improves ICL. Different from prior work, we address multi-modal in-context learning of object detection.

Prompt Optimization seeks to automate prompt engineering to maximize model performance on downstream tasks [48]. This problem has been extensively studied for LLMs, where prompting provides a natural and flexible interface for humans to interact with generalist models. Notably, prompting has become a standard approach for solving many flagship NLP tasks [8, 51, 52]. However, since LLMs are particularly sensitive to user prompts [60], they often require careful prompt design [49, 54]. Soft prompt tuning methods demonstrate strong performance using gradient-based optimization [28, 36, 46]. However, such optimization techniques are impractical for frontier MLLMs that typically only accessible via APIs, and state-of-the-art open-weights models are prohibitively expensive to fine-tune on consumer-grade hardware. In this work, we draw inspiration from discrete prompt search methods such as prompt generation [2, 6, 41], prompt scoring [17], and prompt paraphrasing [21, 68] to optimize class descriptions directly in natural language space.

3 Detection Prompt Optimization for FSOD

In this section, we present our Detection Prompt Optimization (DetPO) framework for improving MLLM alignment to target concepts using few-shot multi-modal examples. DetPO generates initial class descriptions based on ground truth bounding boxes and iteratively refines these descriptions based on model predictions on the training set through contrastive prompt refinement. Importantly, we perform this iterative optimization independently for each class. We describe our algorithm below and provide psuedo-code in Appendix C.

Constrastive Prompt Refinement. Unlike traditional detectors, which are trained primarily on true positive examples, we find that MLLMs achieve higher detection accuracy when provided with corner cases and negative examples, similar to how human annotators learn from examples of what *not* to annotate. For each class C, we generate an initial class description by prompting the model to describe key features of all ground-truth instances of C in the training set. We then refine this description by prompting the model to contrast C with ground-truth instances from all other classes, encouraging the model to focus on discriminative features. Using this detailed prompt, we run inference on the training images to generate candidate detections and identify all false positives and false negatives. Next, we find the top K most severe false positives and false

Fig. 2: Contrastive Prompt Refinement Reduces Class Confusion. At each iteration, we use the current class description to query the MLLM and obtain a set of candidate detections on the training set. From these predictions, we identify true positive, false positive, and false negative detections. The prompt is then refined by asking the MLLM to adjust the class definition such that it explicitly *excludes* false positives and *includes* false negatives. This iterative procedure is repeated until convergence. We illustrate a single refinement step above, where the highlighted text indicates newly added details that encode the desired correction differentiating `Serve` from `Attack`.

negatives (e.g., the false positives with the highest confidence scores and the false negatives with the lowest IoU), and use these predictions to guide the next refinement step. For one sampled false positive prediction, we instruct the model to revise the class definition to explicitly *exclude* that incorrect example. For one sampled false negative, we ask the model to revise the definition to explicitly *include* the missed instance. We provide the corresponding ground-truth instance to ground the correction. To better capture contextual cues, we prompt the model with full images and highlight referred instances with bounding boxes. After updating the prompt, we rerun inference on the few-shot training set and repeat this process until training set performance converges or a predefined maximum number of refinement steps is reached (Figure 3).

Confidence Score Estimation. Although specialist detectors typically predict per-box confidence scores, MLLMs do not by default. We find that simply prompting the model to predict a confidence score per-box works well in practice, down weighting false positives more effectively than the baseline prompt. We include all prompt templates in Appendix D.

VQA Score. Directly asking the model to self-report bounding box confidence scores inside the optimization loop effectively balances iteration speed

and confidence estimation accuracy. However, it does not explicitly incorporate object-specific visual features. To address this limitation, we post-process the confidence scores after optimization for each predicted bounding box on the test set using VQA Score [33]. Specifically, for each prediction, we draw a box on the original image and prompt the model with the binary question: "Is there an instance of class `{CLS}` inside this bounding box? Please answer Yes or No." We compute the final confidence score as the normalized likelihood of the `Yes` token (i.e., $\frac{p(\texttt{Yes})}{p(\texttt{Yes})+p(\texttt{No})}$). Although VQA Score provides higher quality confidence estimates than self-reported scores, its runtime scales linearly with the number of predictions, making it computationally expensive in practice. We include this optional re-ranking step as an upper bound on the performance of our approach, but note that using self-reported confidence scores is a strong alternative (Table 3). Notably, recent black-box LLMs like Gemini 3 Pro [13] do not expose token probabilities in their API, likely to hamper efforts to distill the model's predictions. Therefore, we utilize Qwen3-VL (30B-A3B) to post-process Gemini's predictions to generate VQA-based confidence scores.

4 Experiments

In this section, we evaluate DetPO against recent state-of-the-art specialist and generalist detectors. Further, we ablate our design choices to highlight the impact of contrastive prompt refinement and confidence score estimation.

Datasets and Metrics. We evaluate DetPO on Roboflow20-VL (RF20-VL), a subset of the Roboflow100-VL [45] benchmark. RF20-VL is a few-shot object detection benchmark curated from Roboflow Universe, a community-driven platform that hosts diverse open-source datasets for real-world computer vision tasks. Notably, the benchmark includes 20 datasets spanning diverse domains such as aerial imagery, X-rays, medical imaging, and wildlife monitoring. Each dataset provides 10-shot training examples and rich annotation instructions per class. We follow the standard COCO evaluation protocol [32] and report mean Average Precision (mAP) for each super-category, as defined by Robicheaux et al. [45], along with the average mAP across all 20 datasets. We refer readers to Appendix A for full details of our experimental setup, model configurations, and prompts, and Appendix E for benchmarking results on LVIS.

Baselines. We evaluate the zero-shot performance of specialist detectors like GroundingDINO [34], LLMDet [18], SAM3 [9], MQ-GLIP [65], and YOLO-E [58], as well as generalist models like Qwen 2.5-VL [5], Qwen 3-VL [4], and Gemini 3 Pro [13]. Further, we evaluate prompt optimization approaches like GEPA [2] and MIPROv2 [41] with Qwen 3-VL and Gemini 3 Pro using DSPy [23].

DetPO Outperforms Specialist Zero-Shot Models. While specialist detectors outperform baseline generalist models (with the exception of Gemini 3 Pro) on RF20-VL, prompting generalist models with DetPO's black-box optimized prompts significantly outperforms specialist object detectors (Table 2). While top-performing specialist models like LLMDet and GroundingDINO achieve 17.2 and 16.8 AP, respectively, their test-time performance is fundamentally limited

Table 2: Roboflow20-VL Benchmark. We evaluate recent specialist and generalist models on 20 datasets from the Roboflow20-VL (RF20-VL) benchmark. Notably, we find that specialist object detectors such as MQ-GLIP and YOLO-E fail to benefit from few-shot visual examples, with performance capped at 14.0 mAP. The best specialist model, LLMDet (17.2 mAP), relies solely on class name inputs. In contrast, generalist MLLMs like Qwen3-VL (30B-A3B) and Gemini 3 Pro augmented with DetPO outperform specialist detectors, reaching 21.6 and 26.4 mAP respectively. Across all evaluated generalist models, our Detection Prompt Optimization (DetPO) framework substantially improves accuracy over the baseline. Note that C is class names, I is instructions, and V is for images.

Method	Aerial	Document	Flora & Fauna	Industrial	Medical	Sports	Other	All
Specialist Models								
GroundingDINO [34] (C)	28.5	5.1	33.7	12.8	0.4	5.1	16.9	16.8
LLMDet [18] (C)	32.3	4.4	33.6	12.6	0.7	6.7	16.7	17.2
SAM3 [9] (C)	32.3	15.3	17.1	13.8	2.0	14.2	17.8	16.3
MQ-GLIP [65] (C)	30.1	2.5	32.8	5.5	0.5	6.4	10.8	14.0
MQ-GLIP [65] (V)	1.8	1.1	17.6	1.8	0.1	6.6	6.8	6.7
MQ-GLIP [65] (C + V)	29.8	2.5	32.7	5.6	0.5	6.5	10.9	14.0
YOLO-E [58] (C	10.2	1.6	16.4	8.1	0.3	7.8	10.9	9.2
YOLO-E [58] (V)	11.6	10.7	17.5	15.2	2.3	8.5	16.4	13.2
YOLO-E [58] (C + V)	12.8	6.1	21.0	14.7	1.7	10.9	15.7	13.4
Generalist Models								
Qwen3-VL [4] (30B-A3B) (C + I)	9.0	7.8	23.5	9.6	0.7	14.4	10.1	11.9
w/ DetPO (Ours) + VQA Score [33]	16.1	25.2	36.5	20.1	0.2	25.7	18.4	21.6
w/ GEPA [2]	9.3	12.4	23.6	10.8	1.3	15.1	11.3	13.0
w/ MIPROv2 [41]	8.7	5.6	18.6	10.3	0.0	15.1	9.9	10.7
Gemini 3 Pro [13] (C + I + V)	27.0	26.7	31.3	26.2	2.6	26.9	13.3	23.8
w/ DetPO (Ours) + VQA Score [33]	26.2	35.7	35.4	23.3	3.9	28.2	20.4	26.3
w/ GEPA [2]	19.2	30.6	32.1	32.7	2.0	28.2	20.8	25.6
w/ MIPROv2 [41]	22.9	27.0	31.6	26.4	3.0	29.7	19.8	25.0

because it is difficult to effectively leverage few-shot visual examples. In contrast, applying DetPO to generalist models yields substantial gains. Gemini 3 Pro with DetPO prompts achieves a state-of-the-art 26.4 mAP, outperforming the best specialist model by 9.2%. Similarly, Qwen3-VL (30B-A3B) prompted with our optimized prompt achieves 21.6 mAP, improving by 9.7% over the baseline prompt (Figure 3). This suggests that effectively prompted generalist models can surpass purpose-built zero-shot specialists.

DetPO Improves over Prior Prompt-Optimization Frameworks. Table 2 also shows that DetPO outperforms existing general-purpose prompt optimization techniques like GEPA [2] and MIPROv2 [41] for object detection. When applied to Qwen3-VL (30B-A3B), GEPA provides a marginal improvement over the baseline, achieving 13.0 mAP, while MIPROv2 actually degrades to 10.7 mAP. In contrast, DetPO substantially improves the model's performance to 19.4 AP. This trend also holds for Gemini 3 Pro, where DetPO (26.4 mAP) outperforms GEPA (25.6 mAP) and MIPROv2 (25.0 mAP). These results indicate that our task-specific optimization approach is better suited for extracting robust object detection capabilities from generalist MLLMs than prior approaches.

DetPO Transfers Across Different MLLMs. We evaluate the generalizability of our prompt optimization approach across a diverse set of generalist detectors, including the Qwen2.5-VL and Qwen3-VL families. As shown in Table 3,

Table 3: DetPO Transfers Across Generalist Detectors. We evaluate our detection prompt optimization approach across popular MLLMs like Qwen 2.5-VL and Qwen 3-VL. Notably, we find that DetPO consistently improves over the baseline. Further, VQA Score yields modest improvements for all models, highlighting the importance of well-calibrated confidence score estimates.

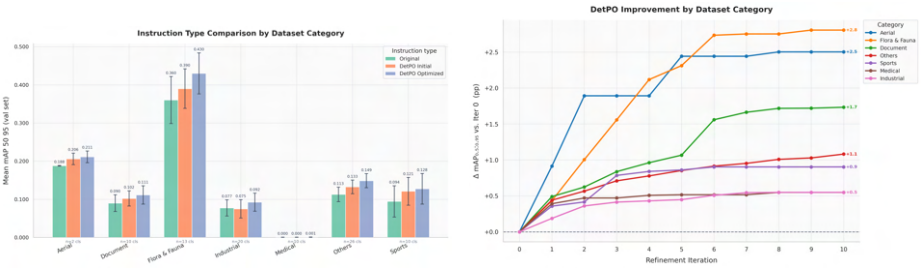
Method	A	D	F & F	I	M	S	O	All
Qwen2.5-VL (7B) [5] (C + I)	4.9	5.1	13.5	3.9	0.1	7.3	5.0	6.2
w/ DetPO (Ours)	6.2	12.1	19.3	4.2	0.0	9.1	7.5	9.1
+ VQA Score [33]	9.4	17.3	23.4	7.2	0.0	12.8	8.8	11.9
Qwen2.5-VL (72B) [5] (C + I)	6.3	10.7	19.0	7.5	0.4	14.4	9.1	10.4
w/ DetPO (Ours)	11.1	23.0	26.1	12.4	0.5	14.8	14.9	15.7
+ VQA Score [33]	10.8	26.3	26.7	13.0	0.5	16.7	15.0	16.5
Qwen3-VL [4] (8B) (C + I)	7.1	7.3	24.8	9.3	0.2	10.2	10.9	11.4
w/ DetPO (Ours)	8.3	19.1	30.3	13.7	0.1	14.2	12.2	15.3
+ VQA Score [33]	12.3	24.2	32.3	13.5	0.2	17.2	14.3	17.5
Qwen3-VL [4] (30B-A3B) (C + I)	9.0	7.8	23.5	9.6	0.7	14.4	10.1	11.9
w/ DetPO (Ours)	13.8	18.6	34.6	19.7	0.1	21.8	16.4	19.4
+ VQA Score [33]	16.1	25.2	36.5	20.1	0.2	25.7	18.4	21.6

Table 4: Ablation on Confidence Score. We evaluate different approaches for estimating confidence scores with Qwen3-VL (30B-A3B). By default, we use the model's self-reported score (row 2). Notably, we find that using SigLIPv2's cosine similarity score between image crops of predicted boxes and class names for confidence scoring degrades overall performance (dropping from 19.4 to 16.4 mAP). In contrast, VQA Score effectively calibrates detection confidence, improving performance to 21.6 mAP.

Method	A	D	F & F	I	M	S	O	All
Qwen3-VL [4] (30B-A3B) (C + I)	9.0	7.8	23.5	9.6	0.7	14.4	10.1	11.9
+ Contrastive Prompt Optimization	13.8	18.6	34.6	19.7	0.1	21.8	16.4	19.4
w/ SigLIPv2 [57] Score	14.1	13.4	28.0	18.8	0.0	17.3	14.0	16.4
w/ VQA Score [33]	16.1	25.2	36.5	20.1	0.2	25.7	18.4	21.6

DetPO consistently improves detection performance over baseline prompts, regardless of model scale or architecture. For example, smaller models like Qwen2.5-VL (7B) improve from 6.2 to 9.1, while larger open models like Qwen2.5-VL (72B) improve from 10.4 to 15.7 mAP. Furthermore, VQA Score yields modest but consistent improvements across Qwen model variants (e.g., boosting Qwen3-VL 8B from 15.3 to 17.5 mAP), suggesting that precise confidence estimation is critical for maximizing detection performance in generalist models.

Contrastive Prompt Optimization Reduces Class Confusion. We visualize a detection confusion matrix [44] for Qwen3-VL (30B-A3) in Figure 4. Based on the confusion matrices, DetPO and VQA Score generally improve the model's ability to distinguish nuanced or underrepresented classes. The most dramatic improvement occurs in the Wb-Prova dataset, where the baseline almost entirely fails to identify `Juvenile` and `Piglet` boars, instead misclassifying them as `Adult`. Further, VQA Score sigificantly improves their true positive rates on the diagonal to 63% and 79% respectively, effectively resolving the severe class imbalance. In the Actions dataset, the baseline struggles with specific actions like `Block` (16%), `Defense` (45%), and `Serve` (22%). Adding DetPO slightly improves `Block` (32%) and achieves near-perfect `ball` recall (99%), while



(a) Instruction Type Comparision (b) Iterative Accuracy Improvement

Fig. 3: Improvement from Contrastive Prompt Refinement. We compare the original baseline prompt against both the initial DetPO prompt and the final optimized DetPO prompt (**left**). These results demonstrate that the DetPO optimized prompts consistently improves detection accuracy across nearly all categories. Further, we show that successive refinement iterations improves performance on the training set (**right**). Importantly, we plot the change in mAP relative to the initial DetPO prompt. Most domains show strong initial gains that begin to plateau around iteration 6, with the Flora & Fauna and Aerial categories showing the largest overall improvements (+2.8 and +2.5, respectively).

VQA Score significantly helps distinguish nuanced actions, bringing `Defense` to 68% and `Serve` to 59%. Finally, in the Defect Detections dataset, the baseline suffers from aggressive overconfidence, misclassifying 98% of `non-defective fishplates` as defective. Incorporating DetPO and VQA Score helps mitigate false positives, incrementally improving the model's ability to correctly recognize non-defective parts (up to 12%). We include examples of the original prompt, DetPO's initial prompt, an DetPO's optimized prompt for the `soft plastic` class below. We highlight key details from each prompt in blue.

Original Prompt: Soft plastic is often transparent or semi-transparent, featuring a flexible, wrinkled appearance, and have diverse visual appearances.

DetPO Initial Prompt: 'Soft plastic': Small, flexible, and thin sheets or bags made of translucent plastic material. Characterized by a smooth, shiny surface that reflects light, often with crinkled or folded textures.

DetPO Optimized Prompt: 'Soft plastic': Thin, flexible, and translucent plastic material that appears in crumpled, or folded forms, often as sheets, bags, or loose fragments. Characterized by a smooth, shiny surface that reflects light, with crinkled or wrinkled textures. It may hold contents but lacks rigid structures, doesn't maintain fixed shape, and conforms to surfaces.

Fig. 4: Detection Confusion Matrix. We compare Qwen3-VL (30B-A3B), Qwen3-VL with DetPO, and with VQA Score across the Actions, Wb-Prova, and Defect Detections datasets. We find that DetPO and VQA Score consistently resolve baseline class imbalances. Notably, our proposed approach improves true positive rates for underrepresented classes (`Juvenile`, `Piglet`) and nuanced actions (`Defense`, `Serve`), while mitigating aggressive false positive predictions in defect detection.

Estimating Confidence Scores Improves Detection Accuracy. Table 4 evaluates different methods for confidence estimation applied to Qwen3-VL (30B-A3B). Our results suggest that self-reported confidence scores (row 2) provide a significant improvement over the baseline, which does not predict per-box confidences at all (where we assign each bounding box with a score of 1.0). We find that using SigLIPv2's cosine similarity score between image crops of predicted boxes and class names for confidence scoring degrades overall performance, dropping from 19.4 to 16.4 mAP, particularly hurting performance for Documents (18.6 to 13.4 mAP) and Flora & Fauna (34.6 to 28.0 mAP). In contrast, we find that VQA Score proves to be highly effective; by leveraging the model's own logit scores, we can boost overall performance to 21.6 mAP. Furthermore, VQA Score demonstrates consistent improvements across all domains compared to our self-reported scoring approach.

Table 5: Black Box Prompting vs White-Box Fine-Tuning. We compare black box prompting methods against fine-tuned open-weight models. While our DetPO framework significantly improves the spatial reasoning capabilities of frontier MLLMs like Gemini 3 Pro, full white-box fine-tuning of specialist models (e.g., GroundingDINO) currently performs the best. However, our analysis reveals a strong trendline as recent closed-source MMLMs like Gemini are far more performant. We suspect that DetPO applied to the next generation of frontier MMLMs will outperform all prior art.

Method	A	D	F & F	I	M	S	O	All
Black Box Prompting								
GroundingDINO [34] (C)	28.5	5.1	33.7	12.8	0.4	5.1	16.9	16.8
Qwen3-VL [4] (8B) (C + I)	7.1	7.3	24.8	9.3	0.2	10.2	10.9	11.4
w/ DetPO (Ours) + VQA Score	12.3	24.2	32.3	13.5	0.2	17.2	14.3	17.5
Qwen3-VL [4] (30B-A3B) (C + I)	7.1	7.3	24.8	9.3	0.2	10.2	10.9	11.4
w/ DetPO (Ours) + VQA Score	16.1	25.2	36.5	20.1	0.2	25.7	18.4	21.6
Gemini 3 Pro [13] (C + I + V)	27.0	26.7	31.3	26.2	2.6	26.9	13.3	23.8
w/ DetPO (Ours) + VQA Score	26.2	35.7	35.4	23.3	3.9	28.2	20.4	26.3
White-Box Fine-Tuning								
Grounding-DINO (Fine-Tuned) [45]	39.9	34.5	45.7	37.8	23.3	26.3	24.7	33.4
Qwen3-VL [4] (8B) (C + I) (LoRA)	7.9	13.7	26.2	10.2	0.3	11.0	12.3	13.2

In-Context Prompting Under Performs Fine-Tuning. Table 5 shows that black-box prompt optimization still underperforms "white-box" fine-tuning of open-weight models. We note that this is surprising given that for other recognition tasks like visual question answering (VQA), frontier closed-source models resoundingly outperform their open-weight counterparts [29]. Currently, this is not the case for spatial tasks such as object detection. However, our analysis reveals a strong trendline as recent closed-source MMLMs like Gemini are far more performant. We suspect that DetPO applied to the next generation of frontier MMLMs will outperform fine-tuned open-weight models such as GroundingDINO. We finetune GroundingDINO with mmdetection [10] and Qwen3-VL with LLaMaFactory [69].

Analysis of DetPO Errors. We systematically analyze model errors using TIDE [7] in Figure 5. Our analysis shows that DetPO directly addresses the baseline model's primary weakness by substantially reducing the class confusion rate. We attribute this improvement to contrastive prompt refinement, which explicitly highlights discriminative features between classes. However, re-scoring detections with VQA Score creates a new trade-off: while classification errors remain low and false positives decrease overall, localization errors and missed detections increase substantially. We posit that this is because some true positive detections are incorrectly down-weighted. Overall, DetPO effectively reduces both false positives and mis-classifications, significantly improving detection accuracy.

Qualitative Examples. We visualize qualitative examples in Figure 6. Our approach significantly outperforms the baseline Qwen3-VL model across a diverse set of challenging domains, including aerial imagery, sports, agriculture, thermal imaging, and underwater scenes. The baseline model generates many dense, overlapping, and erroneous bounding boxes in both the aerial airplane scene and the thermal street view, leading to many false positives. Further, it suffers from poor recall, failing almost entirely to detect the `wheat heads`. In contrast,

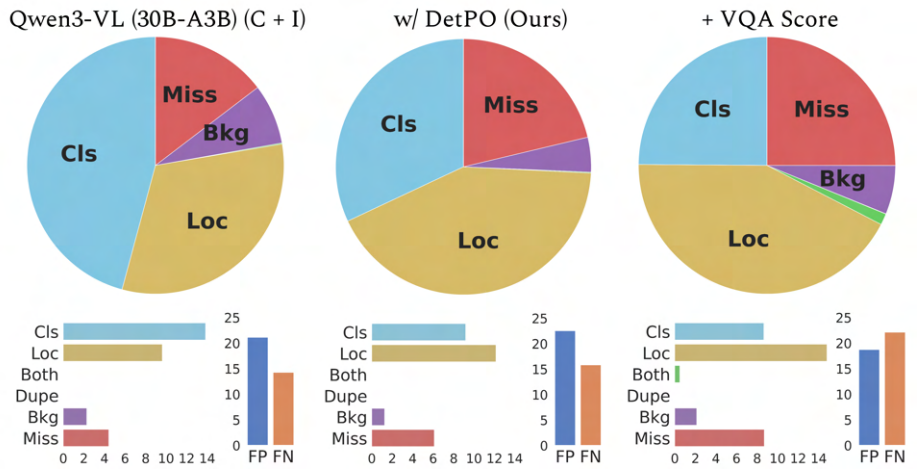


Fig. 5: Detection Errors. We diagnose errors in the baseline Qwen3-VL (30B-A3B) model (**left**), the proposed DetPO method (**center**), and DetPO + VQA Score (**right**) with TIDE [7]. The top row shows the relative distribution of error types, while the bottom row describes the absolute error counts and the overall false positive (FP) versus false negative (FN) rates. DetPO notably reduces classification errors compared to the baseline. While adding in VQA Score successfully reduces overall false positives, it shifts the primary error bottleneck to localization (Loc) and significantly increases missed detections (FN).

the proposed method effectively mitigates these shortcomings, suppressing false positives to produce precise, well localized predictions that closely align with the ground truth. It also successfully recovers objects missed by the baseline (such as the `wheat heads` and smaller `fish`), demonstrating strong performance across diverse environments.

Limitations and Future Work While Detection Prompt Optimization (DetPO) effectively bridges the gap between zero-shot generalization and task-specific adaptation, the iterative prompt refinement process introduces computational overhead similar to training a specialist model. However, it is important to note that this optimization cost is only incurred once during the initial prompt discovery phase; subsequent inference calls using the optimized prompt are no more expensive than standard MLLM inference calls. We expect that future innovations in speeding up inference (e.g. vLLM [27]) will further reduce wall-clock time. Furthermore, our experiments demonstrating the effectiveness of VQA Score suggest that MLLM self-reported confidence scores are suboptimal. Future work may improve performance further by investigating better confidence calibration mechanisms. Finally, evaluating closed-source MLLMs like Gemini 3 Pro via APIs introduces potential training data leakage concerns. By evaluating these models on specific training examples, there is a risk of images and optimized prompts being used in future pre-training mixtures.

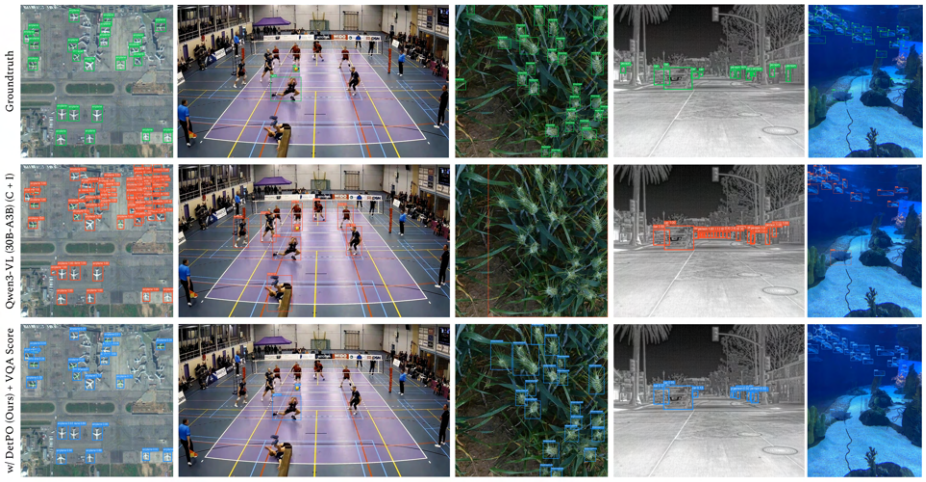


Fig. 6: Qualitative Results. The baseline Qwen3-VL model suffers from high false positive rates (dense, overlapping boxes) and poor recall in complex environments. In contrast, our proposed method mitigates these issues, significantly reducing erroneous predictions while successfully recovering missed objects (like `wheat heads` and `fish`). Best viewed zoomed in.

5 Conclusion

In this paper, we demonstrate that current MLLMs struggle to effectively leverage few-shot multi-modal in-context examples for object detection. To address this limitation, we present Detection Prompt Optimization (DetPO) as a simple yet effective solution. By iteratively refining text-only prompts and incorporating visual feedback from both positive and negative detections, DetPO bridges the gap between zero-shot generalization and task-specific adaptation without gradient-based training. Our experiments on RF20-VL and LVIS demonstrate that DetPO improves concept alignment, reduces false positive detections through improved confidence calibration, and yields consistent performance improves across diverse domains and generalist MLLMs. Our results suggest that prompt-level optimization can serve as a practical alternative to fine-tuning in cases where it is infeasible (e.g., closed-source APIs) or prohibitively expensive (e.g., state-of-the-art open-weights MLLMs), enabling generalist MLLMs to better adapt to novel classes, modalities, and tasks encountered in real-world scenarios.

Acknowledgments

This work was supported in part by the NSF GRFP (Grant No. DGE2140739).

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023) 1
2. Agrawal, L.A., Tan, S., Soylu, D., Ziems, N., Khare, R., Opsahl-Ong, K., Singhvi, A., Shandilya, H., Ryan, M.J., Jiang, M., et al.: Gepa: Reflective prompt evolution can outperform reinforcement learning. arXiv preprint arXiv:2507.19457 (2025) 5, 7, 8, 21, 31
3. Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al.: Flamingo: a visual language model for few-shot learning. Advances in neural information processing systems **35**, 23716–23736 (2022) 5
4. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., Ge, W., Guo, Z., Huang, Q., Huang, J., Huang, F., Hui, B., Jiang, S., Li, Z., Li, M., Li, M., Li, K., Lin, Z., Lin, J., Liu, X., Liu, J., Liu, C., Liu, Y., Liu, D., Liu, S., Lu, D., Luo, R., Lv, C., Men, R., Meng, L., Ren, X., Ren, X., Song, S., Sun, Y., Tang, J., Tu, J., Wan, J., Wang, P., Wang, P., Wang, Q., Wang, Y., Xie, T., Xu, Y., Xu, H., Xu, J., Yang, Z., Yang, M., Yang, J., Yang, A., Yu, B., Zhang, F., Zhang, H., Zhang, X., Zheng, B., Zhong, H., Zhou, J., Zhou, F., Zhou, J., Zhu, Y., Zhu, K.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025) 1, 3, 4, 7, 8, 9, 12, 24, 31, 34
5. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., et al.: Qwen2. 5-vl technical report. arXiv preprint arXiv:2502.13923 (2025) 1, 3, 4, 7, 9
6. Ben-David, E., Oved, N., Reichart, R.: Pada: Example-based prompt learning for on-the-fly adaptation to unseen domains. Transactions of the Association for Computational Linguistics **10**, 414–433 (2022) 5
7. Bolya, D., Foley, S., Hays, J., Hoffman, J.: Tide: A general toolbox for identifying object detection errors. In: European Conference on Computer Vision. pp. 558–573. Springer (2020) 12, 13, 32
8. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020) 2, 5
9. Carion, N., Gustafson, L., Hu, Y.T., Debnath, S., Hu, R., Suris, D., Ryali, C., Alwala, K.V., Khedr, H., Huang, A., et al.: Sam 3: Segment anything with concepts. arXiv preprint arXiv:2511.16719 (2025) 7, 8, 31
10. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019) 12
11. Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al.: Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261 (2025) 1
12. DeepMind, G.: Introducing gemini 2.0: our new ai model for the agentic era (Dec 2024), https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/ 4

13. DeepMind, G.: Gemini 3 pro model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf (2025) 3, 4, 7, 8, 12
14. Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Chang, B., et al.: A survey on in-context learning. In: Proceedings of the 2024 conference on empirical methods in natural language processing. pp. 1107–1128 (2024) 2, 5
15. Du, Y., Wei, F., Zhang, Z., Shi, M., Gao, Y., Li, G.: Learning to prompt for open-vocabulary object detection with vision-language model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14084–14093 (2022) 4
16. Fan, Q., Zhuo, W., Tang, C.K., Tai, Y.W.: Few-shot object detection with attention-rpn and multi-relation detector. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4013–4022 (2020) 4
17. Feldman, J., Davison, J., Rush, A.M.: Commonsense knowledge mining from pretrained models. arXiv preprint arXiv:1909.00505 (2019) 5
18. Fu, S., Yang, Q., Mo, Q., Yan, J., Wei, X., Meng, J., Xie, X., Zheng, W.S.: Llmdet: Learning strong open-vocabulary object detectors under the supervision of large language models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 14987–14997 (2025) 7, 8, 31
19. Gu, X., Lin, T.Y., Kuo, W., Cui, Y.: Open-vocabulary object detection via vision and language knowledge distillation. arXiv preprint arXiv:2104.13921 (2021) 4
20. Gu, X., Lin, T.Y., Kuo, W., Cui, Y.: Open-vocabulary object detection via vision and language knowledge distillation. arXiv preprint arXiv:2104.13921 (2021) 4
21. Jiang, Z., Xu, F.F., Araki, J., Neubig, G.: How can we know what language models know? Transactions of the Association for Computational Linguistics **8**, 423–438 (2020) 5
22. Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., Darrell, T.: Few-shot object detection via feature reweighting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8420–8429 (2019) 4
23. Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., Haq, S., Sharma, A., Joshi, T.T., Moazam, H., et al.: Dspy: Compiling declarative language model calls into self-improving pipelines. arXiv preprint arXiv:2310.03714 (2023) 7, 20
24. Khurana, M., Peri, N., Ramanan, D., Hays, J.: Shelf-supervised multi-modal pre-training for 3d object detection. arXiv preprint arXiv:2406.10115 (2024) 4
25. Köhler, M., Eisenbach, M., Gross, H.M.: Few-shot object detection: A comprehensive survey. arXiv preprint arXiv:2112.11699 (2021) 4
26. Kuo, W., Cui, Y., Gu, X., Piergiovanni, A., Angelova, A.: F-vlm: Open-vocabulary object detection upon frozen vision and language models. arXiv preprint arXiv:2209.15639 (2022) 4
27. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C.H., Gonzalez, J.E., Zhang, H., Stoica, I.: Efficient memory management for large language model serving with pagedattention. In: Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (2023) 13
28. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691 (2021) 5
29. Li, B., Lin, Z., Peng, W., et al.: Naturalbench: Evaluating vision-language models on natural adversarial samples. In: The Thirty-eighth Annual Conference on Neural Information Processing Systems (2024) 12
30. Li, C., Liu, H., Li, L.H., Zhang, P., Aneja, J., Yang, J., Jin, P., Hu, H., Liu, Z., Lee, Y.J., Gao, J.: Elevater: A benchmark and toolkit for evaluating language-augmented visual models. Neural Information Processing Systems (2022) 1, 4

31. Li, L.H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., Wang, L., Yuan, L., Zhang, L., Hwang, J.N., et al.: Grounded language-image pre-training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10965–10975 (2022) 4
32. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13. pp. 740–755. Springer (2014) 7
33. Lin, Z., Pathak, D., Li, B., Li, J., Xia, X., Neubig, G., Zhang, P., Ramanan, D.: Evaluating text-to-visual generation with image-to-text generation. In: European Conference on Computer Vision. pp. 366–384. Springer (2024) 3, 7, 8, 9, 34
34. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Jiang, Q., Li, C., Yang, J., Su, H., 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, 7, 8, 12, 31
35. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. arXiv preprint arXiv:2303.05499 (2023) 4, 20, 31
36. Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., Tang, J.: Gpt understands, too. AI Open **5**, 208–215 (2024) 5
37. Ma, Y., Peri, N., Wei, S., Hua, W., Ramanan, D., Li, Y., Kong, S.: Long-tailed 3d detection via 2d late fusion. arXiv preprint arXiv:2312.10986 (2023) 4
38. Madan, A., Peri, N., Kong, S., Ramanan, D.: Revisiting few-shot object detection with vision-language models. Advances in Neural Information Processing Systems **37**, 19547–19560 (2024) 2
39. Minderer, M., Gritsenko, A., Houlsby, N.: Scaling open-vocabulary object detection. arXiv preprint arXiv:2306.09683 (2023) 4
40. Minderer, M., Gritsenko, A., Stone, A., Neumann, M., Weissenborn, D., Dosovitskiy, A., Mahendran, A., Arnab, A., Dehghani, M., Shen, Z., et al.: Simple open-vocabulary object detection. In: European Conference on Computer Vision. pp. 728–755. Springer (2022) 4
41. Opsahl-Ong, K., Ryan, M.J., Purtell, J., Broman, D., Potts, C., Zaharia, M., Khattab, O.: Optimizing instructions and demonstrations for multi-stage language model programs. arXiv preprint arXiv:2406.11695 (2024) 5, 7, 8, 21, 31
42. Osep, A., Meinhardt, T., Ferroni, F., Peri, N., Ramanan, D., Leal-Taixe, L.: Better call sal: Towards learning to segment anything in lidar. In: ECCV (2024) 4
43. Peri, N., Dave, A., Ramanan, D., Kong, S.: Towards long-tailed 3d detection (2023) 4
44. Peri, N., Dave, A., Ramanan, D., Kong, S.: Towards long-tailed 3d detection. In: Conference on Robot Learning. pp. 1904–1915. PMLR (2023) 9
45. Popov, M., Robicheaux, P., Madan, A., Robinson, I., Nelson, J., Ramanan, D., Peri, N.: Roboflow100-vl: A multi-domain object detection benchmark for vision-language models. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2025) 2, 7, 12
46. Qin, G., Eisner, J.: Learning how to ask: Querying lms with mixtures of soft prompts. arXiv preprint arXiv:2104.06599 (2021) 5
47. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PmLR (2021) 4

48. Ramnath, K., Zhou, K., Guan, S., Mishra, S.S., Qi, X., Shen, Z., Wang, S., Woo, S., Jeoung, S., Wang, Y., et al.: A systematic survey of automatic prompt optimization techniques. arXiv preprint arXiv:2502.16923 (2025) 5
49. Reynolds, L., McDonell, K.: Prompt programming for large language models: Beyond the few-shot paradigm. In: Extended abstracts of the 2021 CHI conference on human factors in computing systems. pp. 1–7 (2021) 5
50. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**, 211–252 (2015) 4
51. Sanh, V., Webson, A., Raffel, C., Bach, S.H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T.L., Raja, A., et al.: Multitask prompted training enables zero-shot task generalization. arXiv preprint arXiv:2110.08207 (2021) 5
52. Schick, T., Schütze, H.: Exploiting cloze-questions for few-shot text classification and natural language inference. In: Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume. pp. 255–269 (2021) 5
53. Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al.: Laion-5b: An open large-scale dataset for training next generation image-text models. Advances in neural information processing systems **35**, 25278–25294 (2022) 4
54. Shin, T., Razeghi, Y., Logan IV, R.L., Wallace, E., Singh, S.: Autoprompt: Eliciting knowledge from language models with automatically generated prompts. arXiv preprint arXiv:2010.15980 (2020) 5
55. Sun, Q., Cui, Y., Zhang, X., Zhang, F., Yu, Q., Wang, Y., Rao, Y., Liu, J., Huang, T., Wang, X.: Generative multimodal models are in-context learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14398–14409 (2024) 5
56. Takmaz, A., Saltori, C., Peri, N., Meinhardt, T., de Lutio, R., Leal-Taixe, L., Osep, A.: Towards Learning to Complete Anything in Lidar. In: International Conference on Machine Learning (ICML) (2025) 4
57. Tschannen, M., Gritsenko, A., Wang, X., Naeem, M.F., Alabdulmohsin, I., Parthasarathy, N., Evans, T., Beyer, L., Xia, Y., Mustafa, B., et al.: Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. arXiv preprint arXiv:2502.14786 (2025) 9
58. Wang, A., Liu, L., Chen, H., Lin, Z., Han, J., Ding, G.: Yoloe: Real-time seeing anything (2025), https://arxiv.org/abs/2503.07465 7, 8
59. Wang, X., Huang, T.E., Darrell, T., Gonzalez, J.E., Yu, F.: Frustratingly simple few-shot object detection. In: International Conference on Machine Learning (ICML) (2020) 4
60. Webson, A., Pavlick, E.: Do prompt-based models really understand the meaning of their prompts? In: Proceedings of the 2022 conference of the north american chapter of the association for computational linguistics: Human language technologies. pp. 2300–2344 (2022) 5
61. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022) 5
62. Wu, A., Han, Y., Zhu, L., Yang, Y.: Universal-prototype enhancing for few-shot object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9567–9576 (2021) 4

63. Xiao, Y., Lepetit, V., Marlet, R.: Few-shot object detection and viewpoint estimation for objects in the wild. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(3), 3090–3106 (2022) 4
64. Xu, J., Le, H., Samaras, D.: Generating features with increased crop-related diversity for few-shot object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19713–19722 (2023) 4
65. Xu, Y., Zhang, M., Fu, C., Chen, P., Yang, X., Li, K., Xu, C.: Multi-modal queried object detection in the wild. Advances in Neural Information Processing Systems **36**, 4452–4469 (2023) 7, 8
66. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. Advances in neural information processing systems **36**, 11809–11822 (2023) 5
67. Yu, L., Poirson, P., Yang, S., Berg, A.C., Berg, T.L.: Modeling context in referring expressions. In: European conference on computer vision. pp. 69–85. Springer (2016) 1, 4
68. Yuan, W., Neubig, G., Liu, P.: Bartscore: Evaluating generated text as text generation. Advances in neural information processing systems **34**, 27263–27277 (2021) 5
69. Zheng, Y., Zhang, R., Zhang, J., Ye, Y., Luo, Z., Feng, Z., Ma, Y.: Llamafactory: Unified efficient fine-tuning of 100+ language models. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations). Association for Computational Linguistics, Bangkok, Thailand (2024), http://arxiv.org/abs/2403.13372 12
70. Zhong, Y., Yang, J., Zhang, P., Li, C., Codella, N., Li, L.H., Zhou, L., Dai, X., Yuan, L., Li, Y., et al.: Regionclip: Region-based language-image pretraining. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16793–16803 (2022) 4
71. Zhou, X., Girdhar, R., Joulin, A., Krähenbühl, P., Misra, I.: Detecting twenty-thousand classes using image-level supervision. In: European Conference on Computer Vision. pp. 350–368. Springer (2022) 4

A Baseline Implementation Details

We present additional implementation details to reproduce our baseline experiments below.

GroundingDINO is a text-promptable vision-language model designed for open-set object detection. It combines the DINO transformer-based object detector with language grounding so that textual queries like "a brown dog" or "traffic light" guide the detection process. The model aligns image regions with text embeddings to produce bounding boxes for objects that match the query, even if those categories were not explicitly included in training. We use GroundingDINO [35] with pretrained weights from mmdetection (MM-GroundingDINO-L*). We prompt the model with all the class names combined into a single prompt. For "white-box" experiments, we fine-tune GroundingDINO on each few-shot dataset for 1000 iterations with a batch size of 4 and a learning rate of 3e-4. We resize all images to (640, 1333) and don't use any additional data augmentations.

MQ-GLIP proposes a learnable module that enables multi-modal prompting. We choose GLIP with a SWIN-L backbone as the underlying detection model for our experiments. We use the model checkpoint trained on Objects365, FourODs, GoldG, and Cap24M. Lastly, we use class names as the text prompts and few-shot visual examples as visual prompts.

YOLO-E presents a unified open-vocabulary framework that integrates reparameterizable region-text alignment (RepRTA) and semantic-activated visual prompt encoding (SAVPE). We select the YOLOv8-based architecture as the underlying detection model. We utilize the model checkpoint pre-trained on large-scale grounding datasets including Objects365 and GoldG. Lastly, we use class names as text prompts and few-shot training images as visual prompts to enable zero-shot detection.

SAM3 is a recent open-vocabulary image and video segmentation model. Unlike traditional segmentation models that require predefined object classes, SAM3 can detect and generate pixel-level masks for any object described by a concept, using prompts such as natural-language text (e.g., "red car"), example images, or interactive clicks. We prompt for each class independently since SAM3 does not natively support multi-class detection.

LLMDet integrates LLM supervision into GroundingDINO's architecture to improve open-vocabulary object detection. Unlike traditional detectors that can only recognize a fixed set of predefined classes, LLMDet leverages the semantic knowledge of LLMs to learn from both region-level descriptions and image-level captions, enabling it to detect objects described in natural language, even if those object categories were not present during training. This language-guided supervision helps the model generalize better to new or rare object categories and perform zero-shot detection.

B Prompt Optimization Baseline Details

We use DSPy [23] to define a object detection program that takes an image and text prompt as input and produces a JSON string of bounding box detections.

The detection prompt is automatically constructed from each dataset's categories and README metadata (i.e., per-class descriptions and annotator instructions), instructing the model to detect all target classes simultaneously. For Gemini 3 Pro, bounding boxes use the native `[ymin, xmin, ymax, xmax]` format normalized to $[0, 1000]$; for Qwen 3-VL, we use `[x1, y1, x2, y2]` in the same coordinate space. We optimize per-image F1 at IoU ≥ 0.5 with greedy matching. The metric provides structured text feedback (e.g. precision, recall, and specific error descriptions) to guide prompt evolution.

GEPA automatically improve prompts or instructions by estimating gradient-like signals from model outputs and feedback to iteratively update prompts. The algorithm evaluates how changes to a prompt affects task performance and then adjusts the prompt in the direction that improves results. This automatically refines prompts for better accuracy, reasoning quality, or task performance while treating the MLLM as a black box. We run GEPA [2] with the `light` budget preset, generating 6 candidate prompts over ∼10 evolutionary trials. GEPA reflects on minibatches of 3 training examples per step using the same base model as the reflection LM (temperature 0.8), and selects the candidate with the highest validation F1.

MiPROv2 is an automated prompt optimization algorithm that improves LLM performance by generating, evaluating, and combining multiple candidate instructions and demonstrations. It searches over different prompt structures such as task instructions, examples, and formatting and selects the best configuration based on validation performance. We run MIPROv2 [41] with the same `light` budget preset as GEPA. Unlike GEPA, MIPROv2 proposes instruction candidates via LLM-based generation rather than evolutionary reflection, and selects among them using Bayesian optimization over validation scores.

C DetPO Implementation Details

We present additional implementation details to reproduce our DetPO experiments below and open source our code on GitHub.

DetPO (Detection Prompt Optimization) is a single-class iterative prompt optimization framework that automatically refines per-class natural-language descriptions for MLLM-based object detection. DetPO uses the MLLM as both a detector and a prompt critic, iteratively improving textual class definitions by reasoning over its own detection errors. Given a set of object classes $\mathcal{C}$, a training split, and a maximum number of iterations T_{max}, DetPO produces a refined natural-language class definition Pc for each class c that maximizes detection performance on a held-out validation set. The full procedure is summarized in Algorithm 1 and is described in detail below.

Stage 1: Initial Prompt Generation. The goal of stage 1 is to bootstrap a class definition from labeled visual examples before running inference on unannotated test images. It consists of two parts:

SummarizePositive. All ground truth instances in the training set containing class c are annotated with green bounding boxes. Next, the MLLM is

prompted with all annotated images and is asked to identify the consistent visual characteristics of the highlighted objects. This produces a concise class definition. A seed description from the dataset README provides a prior that grounds the initial generation:

$$P_c \leftarrow \text{MLLM}\big(\{\,\text{DrawGreen}(x,\, b_c) \mid x \in \mathcal{X}^+(c)\,\}\big)\,. \tag{1}$$

RefineContrastive. For every other class $c^- \neq c$, one example image containing c^- is randomly sampled. That image is annotated with a red bounding box around the c^- instance. We prompt the MLLM with both a negative example from c^- and positive example from class c. The MLLM then identifies the key features distinguishing the object inside the green box from the red box, and produces an updated P_c that excludes c^- while including c:

$$P_c \leftarrow \text{RefineContrastive}\big(P_c,\, \text{DrawGreen}(x_c^+),\, \text{DrawRed}(x_{c^-}^-)\big)\,. \tag{2}$$

This loop iterates over all $|\mathcal{C}| - 1$ negative classes, progressively sharpening P_c to exclude visually similar but semantically distinct objects.

Stage 2: Iterative Error-Driven Refinement. Stage 2 runs a closed-loop optimization in which P_c is evaluated on the training split. We identify the worst false positive (FP) and worst false negative (FN) detections and update P_c to address each error type.

IdentifyErrors. At each iteration t, we run inference on the training split using P_c and compute COCO-style metrics. Per-detection error scores are derived as follows:

False Positive Score. For each predicted box that does not overlap with any ground truth box of class c, the false positive error is proportional to the model's confidence and its overlap with the ground truth boxes of other classes:

$$\varepsilon_{\text{FP}}(d) \;=\; s_d \cdot \max\big(0.2,\; \text{IoU}(b_d, b^*_{\neq c})\big), \tag{3}$$

where s_d is the detection confidence and $b^*_{\neq c}$ is the nearest ground truth box of any class other than c.

False Negative Score. For each ground truth box of class c that is not matched with a prediction, the false negative error reflects how far the best matching prediction falls from a correct detection:

$$\sigma(g) \;=\; \max_d \big[s_d \cdot \text{IoU}(b_d, g)\big], \tag{4}$$

$$\varepsilon_{\text{FN}}(g) \;=\; 1 - \sigma(g). \tag{5}$$

The worst false positive is selected as $\arg\max_d \varepsilon_{\text{FP}}(d)$ and the worst false negative as $\arg\max_g \varepsilon_{\text{FN}}(g)$. To encourage diversity, previously selected images are excluded from the candidate pool. We use the best match (e.g. highest-scoring correct detection), worse false positive (e.g. highest error false positive), and worse false negative (e.g. highest error false negative) as visual evidence.

RefineInclude. The MLLM then receives the best match (**green**) and worst false negative (**blue**) images alongside the current P_c. The MLLM identifies

features shared by both objects and produces a definition that would also detect the blue instance. False negative refinement is applied before false positive refinement within each iteration so that the broadened definition is subsequently tightened:

$$P_c \leftarrow \text{RefineInclude}\big(P_c,\ x^{+}_{\textbf{green}},\ x^{\text{FN}}_{\textbf{blue}}\big)\,. \tag{6}$$

RefineExclude. The MLLM receives the best match (**green**) and worst false positive (**red**) images alongside the updated P_c. The MLLM is asked to identify features that distinguish the two objects and produce a definition that would reject the worse false positive:

$$P_c \leftarrow \text{RefineExclude}\big(P_c,\ x^{+}_{\textbf{green}},\ x^{\text{FP}}_{\textbf{red}}\big)\,. \tag{7}$$

Conservative Update Rule and Early Stopping After each iteration, we compare the mAP of the updated prompt to the previous iteration's training mAP. If performance decreases, P_c is reverted to the previous best prompt (`prev_instructions`). The best global prompt is tracked separately:

$$P_c^{(t)} \leftarrow \begin{cases} P_c^{(t)} & \text{if mAP}(P_c^{(t)}) \geq \text{mAP}(P_c^{(t-1)}) \\ P_c^{(t-1)} & \text{otherwise (revert)} \end{cases} \tag{8}$$

$$P_c^{*} \leftarrow \arg\max_{t}\ \text{mAP}(P_c^{(t)},\ \mathcal{D}_{\text{train}}). \tag{9}$$

The loop terminates at $t = T_{\max}$ or when all sub-sample evaluation scores are already perfect (early stopping).

Stage 3: Validation Set Candidate Selection After $T_{\max}$ iterations, Stage 3 selects the final prompt from a set of candidates by evaluating each on the held-out validation split.

GenerateAlternative. We also ask the MLLM to generate an additional candidate prompt P_c^{alt} to refine the best prompt without any visual examples. This allows the MLLM to remove dataset-specific artifacts:

$$P_c^{\text{alt}} \leftarrow \text{MLLM}(P_c^{*},\ \textit{"Refine for generalization"})\,. \tag{10}$$

Candidate Evaluation and Selection. We evaluate five candidate prompts on $\mathcal{D}_{\text{val}}$ using COCO mAP. The highest-scoring candidate is selected as the final output for class c:

$$P_c^{\text{final}} \leftarrow \arg\max_{p\,\in\,\mathcal{P}}\ \text{mAP}(p,\ \mathcal{D}_{\text{val}}), \tag{11}$$

where $\mathcal{P} = \{P_c^{(0)},\ P_c^{(1)},\ P_c^{*},\ P_c^{(T_{\max})},\ P_c^{\text{alt}}\}$. $P_c^{(0)}$ is the dataset provided seed prompt, $P_c^{(1)}$ is the Stage 1 prompt, P_c^{*} is the best training iteration prompt, $P_c^{(T_{\max})}$ is the final training iteration prompt, and $P_c^{\text{alt}}\}$ is the alternative generated prompt.

Model Specific Details. We conduct all Qwen2.5-VL experiments using the "qwen2.5-vl-72b-instruct" model and "qwen2.5-vl-7b-instruct" model. Similarly, we conduct all Qwen3-VL experiments using the "qwen3-vl-8b-instruct" and "qwen3-vl-30b-a3b-instruct" models. We prompt the model based on guidelines from Qwen's official documentation. For Gemini 3 Pro experiments, we generate initial DetPO prompts using *gemini-3-flash-preview*. Since the Gemini API does not expose token-level log probabilities, we perform score calibration with VQAScore using Qwen3-VL [4] (30B-A3B).

Algorithm 1: DetPO Algorithm

```
# Single-Class Iterative Prompt Tuning
def tune_prompt(classes, train_set, T_max):
    refined_prompt = ""

    # Stage 1: Initial prompt
    P_c = SummarizePositive(c)
    for c_neg in classes:
        if c_neg == c: continue
        P_c = RefineContrastive(P_c, sample_positive(c),
            sample_negative(c_neg))

    # Stage 2: Iterative refinement
    for t in range(T_max):
        FP, FN = IdentifyErrors(ModelDetect(train_set, P_c))

        # Select worst errors
        FP_err = argmax(FP, key="confidence")
        FN_err = argmin(FN, key="IoU")

        # Update prompt
        x_pos = sample_positive(c)
        P_c = RefineExclude(P_c, x_pos, FP_err)
        P_c = RefineInclude(P_c, x_pos, FN_err)

        refined_prompt = P_c

        # Early stop
        if has_converged(EvaluatePrompt(P_c, train_set)):
            break

    # Stage 3: Val set based selection
    P_c_alt = GenerateAlternative(refined_prompt)
    candidates = [P_c_initial, refined_prompt, P_c_alt]
    refined_prompt = argmax(candidates, key=lambda p:
        EvaluatePrompt(p, val_set))

    return refined_prompt
```

D Prompts

We improve Qwen3-VL's base prompt through small-scale validation on multiple datasets and select the best prompt:

System Prompt

```
‘‘You are a helpful assistant capable of object detection.’’
```

Multi-Class Detection Prompt

```
‘‘Locate all of the following objects: {category_prompt} in
 the image and output the coordinates in JSON format like
{{"bbox_2d":[x1,y1,x2,y2], "label":"class_name"}}.‘‘
```

Single-Class Detection Prompt

```
‘‘Locate every {class name} in the image and output the coordinates in
 JSON format.’’
```

Prompting with Rich Textual Instructions

```
‘‘Locate all of the following objects: {category_prompt} in
 the image and output the coordinates in JSON format like
{{"bbox_2d":[x1,y1,x2,y2], "label":"class_name"}}.
 Use the following annotator instructions to improve detection
 accuracy: {instructions}’’
```

We include the rich textual description for all classes when using the multi-class detection prompt. In contrast, we only append the relevant class description when using the single-class detection prompt.

Prompting with Few-Shot Visual Examples

```
“Locate all of the following objects: {category_prompt} (each
of those is a separate class) in the image and output the
coordinates in JSON format like {{"bbox_2d":[x1,y1,x2,y2],
"label":"class_name"}}.“
```

Initial Class Definition Generation Prompt

```
Analyze the following images and describe the subjects or objects
highlighted in green bounding boxes. Identify and summarize the
key visual characteristics that are consistently observed across
these objects. Emphasize the distinctive features that clearly
differentiate this object class from other elements in the scene.
Your goal is to produce a concise, clear, detailed, and
generalizable definition that enables accurate recognition of this
object class in future images and makes it easily distinguishable
from other objects. Do not mention bounding boxes, colors, or any
annotation details in your response.
```

Class Definition Refinement using False-Positive Prompt

```
Analyze the image carefully and identify the key visual differences
between the object shown in the green bounding box and the one shown
in the red bounding box.
Follow the following steps:
Step-1. Describe the distinguishing visual characteristics that set
apart the object in the green bounding box from the object in the
red bounding box.
Step-2. Based on these distinguishing traits, formulate a clear
and descriptive class definition for the object in the green
bounding box. This definition should focus on its unique visual
and contextual features that help differentiate it from the object
in the red bounding box.
Step-3. Compare your new class definition with the existing
definition of the ‘{class_name}‘ class:
Current class definition of the ‘{class_name}‘ class:
{current_instructions}
Step-4. Synthesize both definitions to produce an improved, more
precise descriptive class definition for the ‘{class_name}’ class.
The updated definition should make it easier to accurately identify
true instances of the ‘{class_name}‘ class while reducing false
positives similar to the one seen in the red bounding box.
Note: Do not mention bounding boxes, colors, or image annotations
in your response. The updated class definition should be a textual
description of the ‘{class_name}‘ class objects.
Return the final updated class definition as descriptive text in the
following format:
‘‘‘python {{‘{class_name}’: <updated definition>}}‘‘‘
```

Class Definition Refinement using False-Negative Prompt

```
Analyze the image carefully and identify the key visual similarities
between the object shown in the green bounding box and the one shown
in the blue bounding box.
Follow the following steps:
Step-1. Describe the similar visual characteristics that set apart
the object in the green bounding box from the object in the blue
bounding box.
Step-2. Based on these similarity traits, formulate a clear and
descriptive class definition for the object in the blue bounding
box as well as the object in the green bounding box. This definition
should focus on its unique visual and contextual features that help
identify both instances of the object in the green and blue bounding
boxes.
Step-3. Compare your new class definition with the existing
definition of the '{class_name}' class provided below:
Current class definition of the `{class_name}' class:
{current_instructions}
Step-4. Synthesize both definitions to produce an improved, more
precise descriptive class definition for the '{class_name}' class.
The updated definition should make it easier to accurately identify
all true instances of the '{class_name}' class similar to the one
seen in the blue and green bounding boxes.
Note: Do not mention bounding boxes, colors, or image annotations
in your response. The updated class definition should be a textual
description of the '{class_name}' class objects.
Return the final updated class definition as descriptive text in the
following format:
```python {{`{class_name}': <updated definition>}}```
```
```

Generate Alternative Class Definition

```
Refine the class definition for the ‘{class_name}’ category.
Objective: Produce a concise, precise, and generalizable definition
that enables reliable recognition of ‘{class_name}’ instances across
diverse images. The definition should clearly distinguish this class
from visually or functionally similar object categories.
Current definition: {best_instructions}
Guidelines: - Focus on intrinsic, stable characteristics such
as structure, shape, components, function, and typical physical
configuration. - Ensure the description is detailed enough for
accurate visual identification, yet broadly applicable across
variations. - Do NOT mention bounding boxes, colors, image
annotations, or dataset-specific context. - Avoid referencing
specific images or examples. - Output only a textual class
definition.
Return the result strictly in the following format:
‘‘‘python {{‘{class_name}’: <updated definition>}}‘‘‘
```

DetPO Single Class Detection Prompt

```
Identify and localize all instances of ‘{class_name}’ in the image.
Output Requirements: - Return valid JSON only. Do not include
explanations or extra text.
- Output a ranked list of detections sorted by confidence (highest
first).
- Include at most 20 detections.
- If no objects are detected, return an empty list [].
For each detection, provide:
- "bbox_2d": [x1, y1, x2, y2]
* Pixel coordinates.
* (x1, y1) = top-left corner.
* (x2, y2) = bottom-right corner.
- "label": "{class_name}"
- "score": float confidence score from 0.0 (lowest) to 1.0 (highest).
Additional Constraints:
- Only include detections that clearly correspond to {class_name}.
- Avoid duplicate or highly overlapping boxes for the same object.
- Follow these annotator instructions to improve detection accuracy:
{dataset_instructions}
Return a JSON list in the following format:

[
{
"bbox_2d": [x1, y1, x2, y2],
"label": "{class_name}",
"score": 0.95
}
]
```

Confidence Estimation using VQA Score Prompt

```
Given the ‘{prompt}’ class defined as follows:
{dataset_instructions}
Is the main subject or object being referred to as ‘{prompt}’
located inside the red bounding box in the image? Please answer
Yes or No. Note: The object should be entirely inside the bounding
box, with no part outside, and it must be the only object present
inside - no other objects should appear within the box.
```

E LVIS Rare 50 10-Shot Benchmark Results

To further evaluate few-shot detection performance, we constructed a subset of LVIS by sampling the 50 least frequent classes in the validation set that contain at least 10 examples each. While many specialist detectors report zero-shot performance on the full LVIS dataset, LLMs typically avoid this due to its large number of classes. We find that specialist detectors like GroundingDINO [35], SAM3 [9], and LLMDet [18] consistently outperform our Qwen3-VL (30B-A3B) generalist baseline. Interestingly, zero-shot predictions from SAM3 slightly outperform GroundingDINO fine-tuned on LVIS v1, suggesting that LVIS categories are likely in-distribution for SAM3's PCS training dataset. Furthermore, LLMDet (which shares GroundingDINO's architecture but does not fine-tune on LVIS) achieves significantly lower performance. Consistent with the results in our main paper, DetPO yields a significant 3.2 mAP improvement over the Qwen3-VL baseline. VQA score further improves performance by 3.4 mAP. In contrast, prompt optimization techniques like GEPA and MIPROv2 fail to improve upon the Qwen3-VL's baseline prompt. We attribute this failure to such methods attempting to optimize for all 50 classes in a single prompt.

We further diagnose Qwen3-VL's errors in Figure 7. Comparing the baseline Qwen3-VL (left) to our DetPO method (center) demonstrates a significant reduction in background errors and overall false positives. While this trade-off inherently increases false negatives, our method yields a more balanced distribution of error types. The addition of VQA Score (right) further limits false positives but considerably increases false negatives.

Table 6: LVIS Rare 50 10-Shot Benchmark. We evaluate model performance on the 50 least frequent LVIS validation classes with atleast 10 examples per-class. While specialist models achieve the highest overall mAP, our DetPO approach and VQA Score substantially improve the performance of the generalist Qwen3-VL baseline.

Method	mAP
Specialist Models	
GroundingDINO (Fine-Tuned) [34] (C)	40.3
SAM3 [9] (C)	40.5
LLMDet [18] (C)	27.1
Generalist Models	
Qwen3-VL [4] (30B-A3B) (C + I)	21.9
w/ DetPO (Ours)	25.1
+ VQA Score	28.5
w/ GEPA [2]	21.9
w/ MIPROv2 [41]	21.9

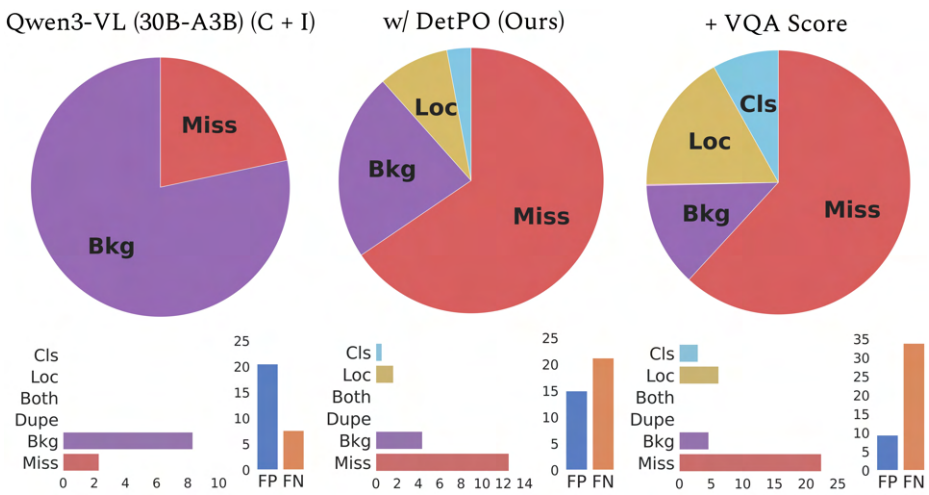


Fig. 7: Detection Errors. We diagnose errors in the baseline Qwen3-VL (30B-A3B) model (**left**), the proposed DetPO method (**center**), and DetPO + VQA Score (**right**) with TIDE [7]. The top row shows the relative distribution of error types, while the bottom row describes the absolute error counts and the overall false positive (FP) versus false negative (FN) rates. DetPO successfully reduces background errors (FP) at the cost of increased misses (FN), a trade-off that is further amplified when incorporating VQA Score.

F Token and Run-Time Analysis

Figures 8 and 9 demonstrate the significant efficiency gains of DetPO over GEPA when evaluated on the aerial-airport dataset. As shown in Figure 8, DetPO achieves an 81% reduction in total token usage. Unlike GEPA, which exhausts the majority of its tokens during the optimization phase, DetPO bypasses this loop and concentrates its token usage almost entirely in the final detection stage. Consequently, this reduction in LLM API calls and token consumption directly translates to faster execution. Figure 9 illustrates that DetPO completes the task in 10 minutes and 46 seconds, making it 1.2× faster than GEPA's 12 minutes and 57 seconds runtime. Ultimately, DetPO offers a streamlined pipeline that conserves both computational resources and real-world processing time. Note that since GEPA optimizes multiple classes in parallel (unlike DetPO, which optimizes each class separately), we expect that DetPO's runtime will increase proportional to the number of classes. Future work should parallelize DetPO's optimization and inference steps to further increase efficiency.

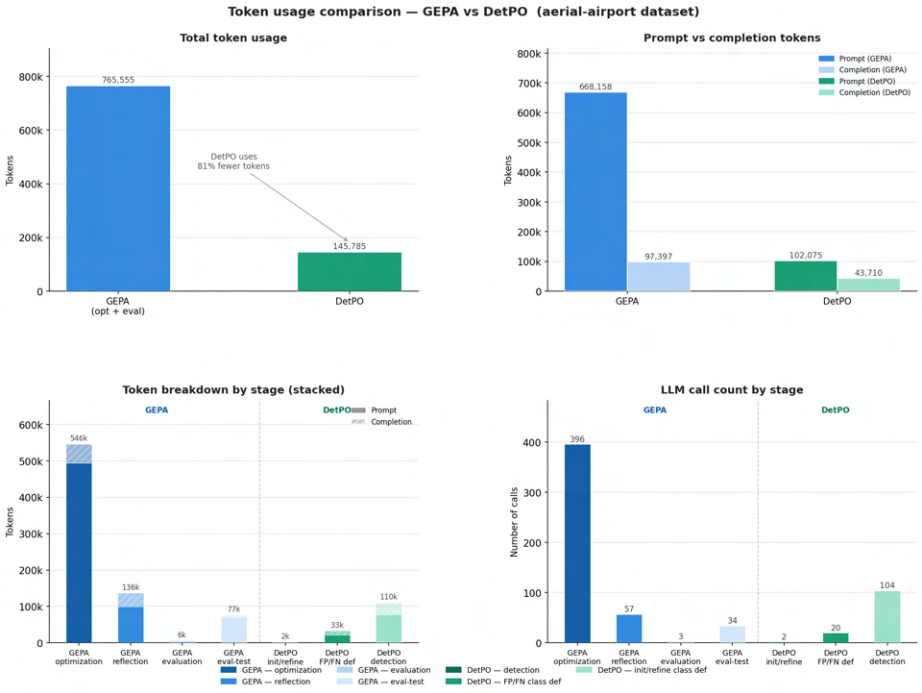


Fig. 8: Token Analysis. We compare token usage between GEPA and DetPO on the aerial-airport dataset. DetPO uses 81% fewer total tokens compared to GEPA (**top left**). DetPO's efficiency gains primarily stem from a massive reduction in prompt tokens (**top right**). In contrast, GEPA expends the vast majority of its tokens (546k) during the optimization phase, whereas DetPO's token usage is concentrated mostly in the final detection stage (**bottom left**). DetPO significantly reduces the number of required API calls compared to GEPA's intensive optimization loop (**bottom right**).

G Impact of K Shots

We evaluate the impact of the number of examples provided to DetPO during optimization. Notably, we find that DetPO's performance (3-shot) improves when given access to additional training examples (5-shot and 10-shot). However, we note that the improvement between 5-shot and 10-shot examples diminishes, suggesting that performance saturates.

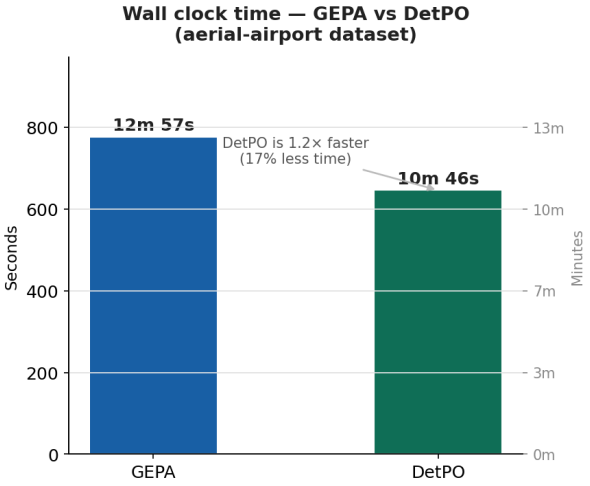


Fig. 9: Wall Clock Analysis. We compare the total execution time between GEPA and DetPO on the aerial-airport dataset. DetPO completes the task in 10 minutes and 46 seconds, outperforming GEPA's time of 12 minutes and 57 seconds. Overall, DetPO achieves a 17% reduction in processing time, making it 1.2x faster than GEPA.

Table 7: Ablation on Few-Shot Examples. We ablate the impact of the number of few-shot examples (3-shot, 5-shot, and 10-shot) using the DetPO on optimization performance. Notably, we see consistent improvements in increasing from 3 shots to 5 shots, but note marginal improvements from 5 shots to 10 shots, suggesting that adding additional examples does not significantly improve model performance.

Method	A	D	F & F	I	M	S	O	All
Qwen3-VL [4] (30B-A3B) (C + I)	9.0	7.8	23.5	9.6	0.7	14.4	10.1	11.9
w/ DetPO 3-shot	12.9	17.2	33.4	19.1	0.1	22.1	16.4	18.9
+ VQA Score	16.2	23.6	33.8	20.1	0.1	26.5	18.3	21.0
w/ DetPO 5-shot	14.1	18.1	34.5	19.0	0.1	22.9	15.6	19.2
+ VQA Score	16.4	25.6	36.6	19.7	0.4	26.1	17.9	21.6
w/ DetPO 10-shot	13.8	18.6	34.6	19.7	0.1	21.8	16.4	19.4
+ VQA Score [33]	16.1	25.2	36.5	20.1	0.2	25.7	18.4	21.6

H Additional Supplemental Materials

We include our code and the final optimized prompts obtained via DetPO on RF20-VL and LVIS Rare 50 in our GitHub repository.