Roboflow100-VL: A Multi-Domain Object Detection Benchmark for Vision-Language Models

Peter Robicheaux[1,*], **Matvei Popov**[1,*], **Anish Madan**[2], **Isaac Robinson**[1], **Joseph Nelson**[1], **Deva Ramanan**[2], **Neehar Peri**[2]
[1] Roboflow, [2]Carnegie Mellon University

rf100-vl.org

Abstract

Vision-language models (VLMs) trained on internet-scale data achieve remarkable zero-shot detection performance on common objects like `car`, `truck`, and `pedestrian`. 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. Rather than simply re-training VLMs on more visual data, we argue that one should align VLMs to new concepts with annotation instructions containing a few visual examples *and* rich textual descriptions. To this end, we introduce Roboflow100-VL, a large-scale collection of 100 multi-modal object detection datasets with diverse concepts not commonly found in VLM pre-training. We evaluate state-of-the-art models on our benchmark in zero-shot, few-shot, semi-supervised, and fully-supervised settings, allowing for comparison across data regimes. Notably, we find that VLMs like GroundingDINO and Qwen2.5-VL achieve less than 2% zero-shot accuracy on challenging medical imaging datasets within Roboflow100-VL, demonstrating the need for few-shot concept alignment. Our code and dataset are available on GitHub and Roboflow.

1 Introduction

Vision-language models (VLMs) trained on web-scale datasets achieve remarkable zero-shot performance on many popular academic benchmarks [55, 27, 46]. However, the performance of such foundation models varies greatly when evaluated in-the-wild, particularly on out-of-distribution classes, tasks (e.g. material property estimation, defect detection, and contextual action recognition) and imaging modalities (e.g. X-rays, thermal spectrum data, and aerial imagery). In this paper, we introduce Roboflow100-VL (RF100-VL), a large-scale multi-domain dataset to benchmark state-of-the-art VLMs on hundreds of diverse concepts not typically found in internet pre-training.

Status Quo. Foundation models are often trained on large-scale datasets curated from diverse sources around the web. However, despite their scale and diversity, these pre-training datasets still follow a long-tail distribution [41], causing foundation models to generalize poorly to rare concepts [36]. A common approach for improving the performance of VLMs is to scale up training data and model size [1]. However, we argue that some data will always remain out-of-distribution, whether due to being sequestered from the internet or being created after the model's training cutoff [48], motivating the need to learn new concepts from a few examples.

Evaluating Out-of-Distribution Generalization. Existing benchmarks primarily assess spatial understanding through visual question answering (VQA) and common sense reasoning [27, 56, 46]. However, we argue that evaluating model performance on compositional reasoning benchmarks alone does not effectively measure generalization to out-of-distribution tasks. Moreover, current spatial understanding and grounding benchmarks (e.g. RefCOCO [54] and OdinW [22]) typically

[*]Equal Contribution

Preprint.

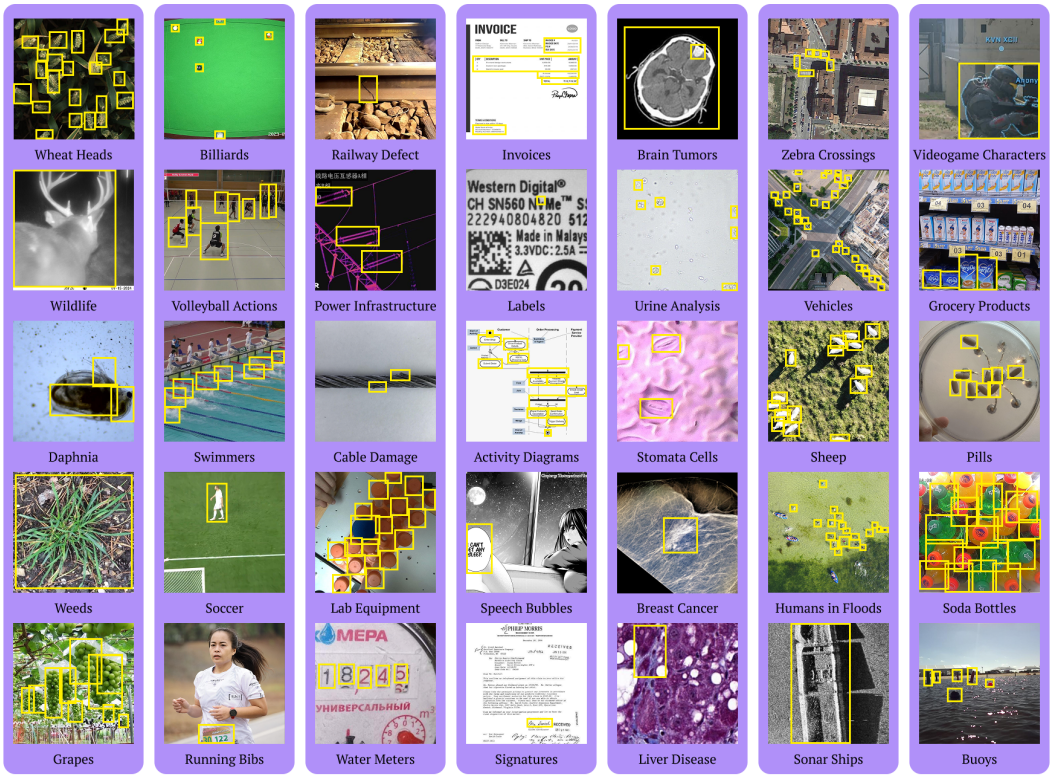
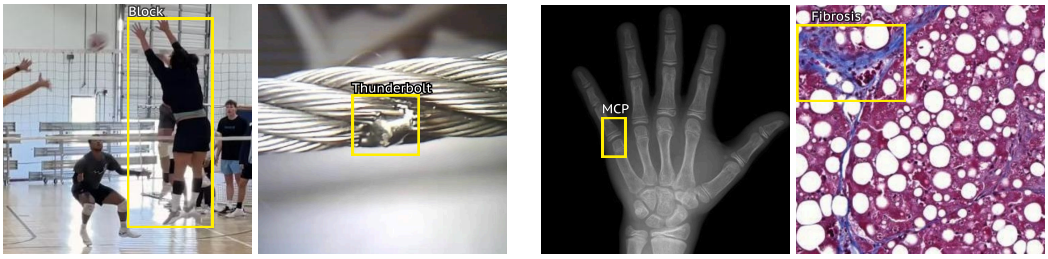


Figure 1: **Roboflow100-VL Dataset.** We identify a set of 100 challenging datasets from Roboflow Universe that contain concepts not typically found in internet-scale pre-training. To simplify analysis, we cluster these 100 datasets using per-dataset CLIP [38] embeddings into seven categories. We visualize examples from each of these categories above. Furthermore, we also generate multi-modal instructions for each dataset with a few visual examples and rich textual descriptions per class to facilitate few-shot concept alignment.

evaluate performance on classes commonly found in internet pre-training. We demonstrate that such benchmarks artificially inflate model performance and are not representative of many real-world applications (cf. Table 1). To address this limitation, we introduce RF100-VL, a large-scale detection benchmark comprised of 100 multi-modal datasets from diverse domains (cf. Fig. 1). Importantly, we carefully curate RF100-VL such that it cannot be solved by simply prompting state-of-the-art models with class names. Specifically, we include datasets where classes are labeled using scientific names (e.g. liver `fibrosis` and `steatosis`), acronyms (e.g. `DIP` and `MCP`), context-dependent names (e.g. detecting a `block` vs. `set` in the context of volleyball), material properties (e.g. `paper` vs. `soft plastic`), and diverse imaging modalities (cf. Fig. 2). We posit that models must leverage multi-modal contextual information (presented in the form of multi-modal annotator instructions) to effectively align to target concepts in RF100-VL.

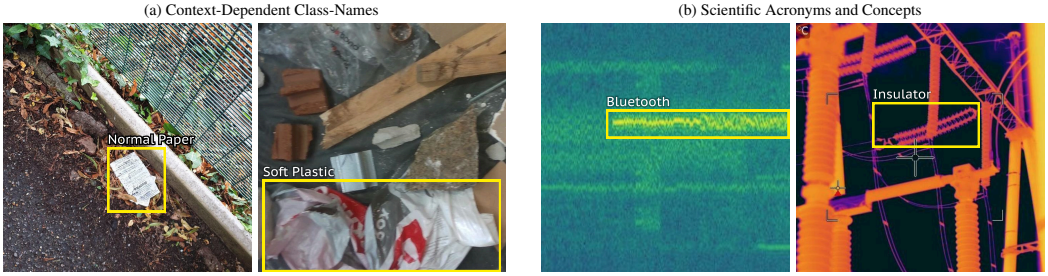
Multi-Modal Annotator Instructions. Annotating large-scale datasets is an iterative process that often requires extensive discussions between data curators and annotators to clarify class definitions and ensure label consistency. These (often multi-modal) labeling instructions provide rich contextual information not provided by class names alone. We argue that aligning foundation models to target concepts can be principally addressed through the lens of few-shot learning by presenting vision-language models with visual examples and rich textual descriptions per class (cf. Fig. 3). Importantly, this approach mirrors how we align human annotators to concepts of interest with few-shot multi-modal examples [4, 30].

Contributions. We present three major contributions. First, we introduce RF100-VL, a large-scale, multi-domain benchmark designed to evaluate vision-language models (VLMs) on challenging real-world use cases. We evaluate state-of-the-art models on our benchmark in zero-shot, few-shot,



(a) Context-Dependent Class-Names

(b) Scientific Acronyms and Concepts



(c) Catch-all Categories

(d) Multi-Modal Imaging Modalities

Figure 2: **Hard Examples in Roboflow100-VL.** Our dataset is particularly challenging because it is difficult to detect objects in RF100-VL using class-names alone. Specifically, we select datasets where classes are labeled using scientific names, acronyms, context-dependent names, material properties. We posit that models must leverage multi-modal contextual annotations to address such hard examples.

semi-supervised, and fully-supervised settings, allowing for comparison across data regimes. Our extensive experiments highlight the difficulty of adapting VLMs to out-of-distribution tasks and reveal the limitations of current state-of-the-art methods. Lastly, we host a challenge at CVPR 2025 in conjunction with the Workshop on Visual Perception via Learning in An Open World to encourage broad community involvement in addressing this challenging problem.

2 Related Works

Vision Language Models are trained using large-scale, weakly supervised image-text pairs sourced from the web. Although many VLMs primarily focus on classification [38] or image understanding, recent methods address spatial understanding with open-vocabulary detectors. Early approaches adapted VLMs for object detection by classifying specific image regions [12, 13] or integrating detection components into frozen [20] or fine-tuned [33, 32, 10] encoders. In contrast, RegionCLIP [59] employs a multi-stage training strategy that involves generating pseudo-labels from captioning data, performing region-text contrastive pre-training, and fine-tuning on detection tasks. GLIP [23] treats detection as a phrase grounding problem by using a single text query for the entire image. Detic [60] improves long-tail detection performance by utilizing image-level supervision from ImageNet [40]. Notably, recent VLMs achieve remarkable zero-shot performance and are widely used as "black box" models in diverse downstream applications [29, 37, 19, 34, 45]. More recently, multi-modal large language models (MLLMs) such as Qwen2.5-VL [2] and Gemini 2.5 Pro [9] frame spatial understanding as a text generation task. Interestingly, such generalist MLLMs perform worse at object detection than task-specific models like GroundingDINO [26]

Fine-Tuning Vision-Language Models is crucial for adapting foundation models to downstream tasks [15, 57, 11]. Traditional fine-tuning methods, such as linear probing [7, 14] and full fine-tuning [50, 51] can be computationally expensive. Instead, parameter-efficient approaches like CLIP-Adapter [11] and Tip-Adapter [58] optimize lightweight MLPs while keeping encoders frozen. Although prior few-shot learners commonly used meta-learning [53], more recent approaches show that transfer learning generalizes better to novel categories [49]. In particular, Pan et. al. [35] demonstrates that transfer learning can be effectively used to fine-tune foundation models using a few multi-modal examples. More recently, in-context learning [52] demonstrates promising results for test-time few-shot adaptation without gradient-based fine-tuning. We explore such test-time fine-tuning strategies in the context of MLLMs [9, 2] to learn from multi-modal annotator instructions.

Figure 3: **Multi-Modal Few-Shot Examples.** We present an example of the few-shot visual examples and rich text descriptions used for in-context prompting and fine-tuning. Notably, image examples used for each class may overlap and are only guaranteed to have exhaustive annotations for one class. Such multi-modal examples help clarify ambiguous concepts like `soft plastic` and `metal`.

Benchmarking Vision-Language Models is of significant interest to the community. State-of-the-art VLMs are typically evaluated using benchmarks such as MMStar [5], MMMU [56], MME [24], ScienceQA [28], MMBench [27], MM-Vet [55], Seed-Bench [21], and MMVP [47]. These benchmarks evaluate a broad set of vision-language tasks, including fine-grained perception, reasoning, common sense knowledge, and problem solving in various domains. However, existing evaluations primarily focus on multi-modal understanding in the context of visual question answering (VQA). In contrast, RF100-VL evaluates VLM detection accuracy given a few visual examples and rich textual descriptions. Prior VLM grounding benchmarks like RefCOCO [54] often focus on referential grounding of common object categories. Recent efforts like ODinW [22] consider more challenging scenarios by sourcing real-world data from Roboflow [8]. However, we find that state-of-the-art methods achieve high zero-shot accuracy on RefCOCO and OdinW [2], suggesting that these datasets may not be well suited for evaluating foundational few-shot object detection [30].

3 Roboflow100-VL Benchmark

As shown in Fig. 1, RF100-VL consists of diverse datasets not typically found in internet-scale pre-training. We highlight our data curation procedure (Section 3.1) and present several baselines to evaluate state-of-the-art models in the zero-shot and few-shot settings (Section 3.2). We also evaluate models under the semi-supervised and fully-supervised settings in Appendix F.

3.1 Creating Roboflow100-VL

We source our datasets from Roboflow Universe, a community-driven platform that hosts diverse open-source datasets created to solve real-world computer vision tasks. With more than $500,000$ public datasets spanning medical imaging, agriculture, robotics, and manufacturing, we focus on selecting high-quality datasets not commonly found in internet-scale pre-training (e.g. COCO [25], Objects365 [42], GoldG [17], CC4M [43]) to better assess VLM generalization to rare concepts. When selecting candidates for RF100-VL, we prioritized datasets where images contained multiple objects, ensuring more realistic evaluation beyond classification. In addition, we sought out datasets with semantically ambiguous names (e.g. "button" can refer to both clothing and electronics) to encourage algorithms to leverage multi-modal annotator instructions rather than simply relying on class names. We manually validate the labeling quality of each dataset to ensure exhaustive annotations. In cases without exhaustive annotations, we manually re-annotate the dataset to the best of our ability (cf. Fig 4). In total, we spent 1693 hours labeling, reviewing, and preparing the dataset.

Multi-Modal Annotation Generation. Annotator instructions offer precise class definitions and visual examples that help clarify annotation policies (e.g. by highlighting typical cases, corner cases, and negative examples) and improve labeling accuracy. Despite providing significant value during

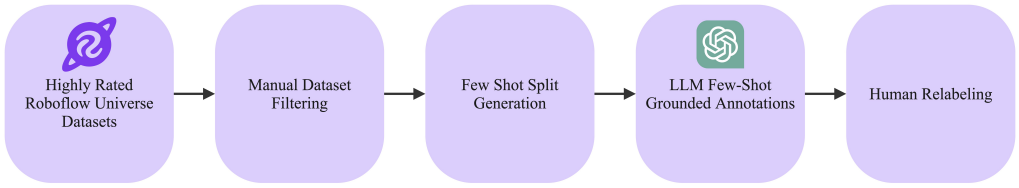


Figure 4: **Dataset Curation.** We begin by sorting all object detection datasets on Roboflow Universe by stars as a proxy for quality and usefulness to the community. Next, we manually filter out all datasets with common classes, datasets where images only have a single focal object, or datasets with watermarks. We generate 10-shot splits following the protocol defined by Wang et.al. [49], where we find a subset of images with 10 total instances per class. We use these 10-shot splits to generate visually grounded "annotator instructions", and manually update these instructions to add any salient details missed by GPT-4o. Finally, human labelers verify that all images within a dataset follow consistent annotation policies (e.g. bounding-box fit, semantic legibility of class names, and completeness of annotation instructions).

the labeling process, few datasets publicly release these annotator instructions. Recognizing the importance of these instructions in aligning humans with target concepts of interest, we generate multi-modal annotator instructions for all 100 datasets within RF100-VL (cf. Fig 3).

We prompt GPT-4o [1] to generate an initial set of annotator instructions, providing in-context examples based on the nuImages annotator guidelines. Our prompt includes a structured output template, along with dataset metadata, class names, and few-shot visual examples per class. In practice, we find that GPT-4o often overlooks the few-shot images and instead relies heavily on class names to generate class descriptions. Notably, GPT-4o struggles when class names are uninformative and sometimes produces overly vague instructions that, while correct, lack useful detail. To address this, we manually verify all generated annotator instructions to mitigate hallucinations and incorporate additional informative visual details missed by the model. We include our annotation generation prompt in Appendix J.

Dataset Statistics. Figure 5 (left) presents an overview of the different types of datasets within RF100-VL, detailing the number of classes, images and annotations per category. RF100-VL contains a total of 564 classes and 164,149 images, with over 1.3 million annotations. The "Other" category has the highest number of classes (142), followed by "Industrial" (122) and "Flora & Fauna" (70). Despite having fewer classes, the "Flora & Fauna" category has the highest number of images (46,718) and annotations (441,677), indicating a higher density of annotations per image. Figure 5 (right) provides a visual representation of class distribution, reinforcing the dominance of the "Other", "Industrial", and "Flora & Fauna" categories. In contrast, "Sports" has the fewest classes (36) and the least representation in RF100-VL. Despite consisting of 100 datasets, RF100-VL has about half the number of images as COCO [25], making this an approachable benchmark for the academic community.

3.2 State-of-the-Art Baselines

We train and evaluate all models on each dataset within RF100-VL independently. Importantly, we do not tune any parameters or modify zero-shot prompts per-dataset. For all models, we compute metrics using pycocotools with maxDets set to 500 instead of the usual 100 because there are many images with more than 100 objects. We discuss our evaluation protocol further in Appendix B.

Zero-Shot Baselines prompt models with class names or expressive descriptions [31] to detect target concepts. However, the effectiveness of zero-shot prompting depends on the pre-training data: If the target class name is semantically meaningful and aligns well with the model's foundational pre-training, performance is strong; otherwise, the model fails catastrophically. We benchmark the zero-shot performance of Detic [60], OWLv2 [32], GroundingDINO [26], MQ-GLIP [52], QwenV2.5-VL [2] and Gemini 2.5 Pro [9].

Few-Shot Baselines. We evaluate three types of few-shot baselines: visual prompting, multi-modal prompting, and federated fine-tuning. Visual prompting uses images of target concepts that are

Dataset Type	# Classes	# Images	# Anno.
Aerial	29	11,627	186,789
Document	88	21,418	127,129
Flora & Fauna	70	46,718	441,677
Industrial	122	29,758	205,627
Medical	77	16,369	125,433
Sports	36	8,443	58,508
Other	142	29,816	210,328
All	564	164,149	1,355,491

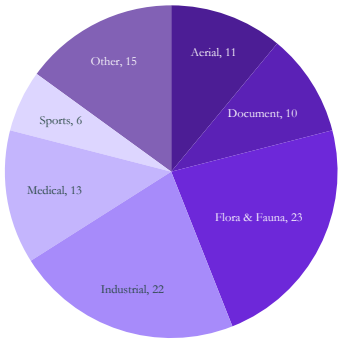


Figure 5: **Dataset Statistics.** The table on the left provides details on the number of classes, images, and annotations across different dataset types within RF100-VL. The figure on the right illustrates the distribution of dataset types by count. Notably, despite containing 100 datasets, RF100-VL is 50% the size of COCO [25] (by number of images) and can feasibly be benchmarked on academic-level compute.

difficult to describe through text as prompts to help models learn novel concepts in-context. For example, while "hard plastic" is a broad and ambiguous category that is hard to define through text, providing image examples improves concept alignment. Typically, visual prompts are tokenized and fed as inputs to a frozen VLM. Here, we apply MQ-GLIP [52] with image prompting. Multi-modal prompting combines language and visual prompts to leverage multi-modal features. Intuitively, using both text and images yields better alignment than using either modality alone. In the case of "soft plastic", ambiguous concepts can be clarified with textual descriptions (e.g., "thin plastic film" and "plastic bag") alongside visual examples. Both visual and language prompts are tokenized and separately fed into a frozen VLM. We evaluate MQ-GLIP [52], and Gemini 2.5 Pro [9] by prompting models with class names, few-shot images, and annotator instructions. Lastly, federated fine-tuning modifies the standard cross-entropy classification to only treat exhaustively annotated classes as true negatives for each image. We follow the implementation from Madan et. al. [30] when fine-tuning Detic [60]. We slightly modify the federated loss when fine-tuning YOLO [16, 18] to avoid using Madan et. al's frequency prior, opting to instead determine hard negatives using per-image annotations.

Table 1: **Comparison to Other Benchmarks.** We find that state-of-the-art MLLMs achieve considerably lower performance on RF100-VL compared to OdinW-13, highlighting the difficulty of our proposed dataset. Further, models that performed better on COCO did not consistently perform better on the RF100-VL, indicating that the newer YOLO models might be overfitting to COCO. Lastly, we highlight a discrepancy between reported and reproduced numbers on both COCO and OdinW. Discrepancies in COCO evaluation can be attributed to differences in evaluation toolkits, while discrepancies in ODinW evaluation can be attributed to prior work evaluating models using referential grounding evaluation protocols, while we use standard object detection evaluation protocols. We discuss this further in section 4.1.

Method	**COCO Val**		**OdinW-13**		**Roboflow100-VL**
	Reported	Ours	Reported	Ours	Ours
Zero-Shot					
Qwen 2.5-VL (7B) [2] (Class Names Only)	-	-	43.1	30.9	7.7
Gemini 2.5 Pro [9] (Class Names Only)	-	-	41.9	33.7	5.6
Fully-Supervised					
YOLOv8n [16]	37.3	37.4	-	-	54.9
YOLOv11n [18]	39.5	39.4	-	-	55.3
YOLOv8s [16]	44.9	45.0	-	-	56.2
YOLOv11s [18]	47.0	46.9	-	-	56.2
YOLOv8m [16]	50.2	50.3	-	-	56.4
YOLOv11m [18]	51.5	51.5	-	-	56.5

4 Experiments

We conduct extensive experiments to evaluate the performance of state-of-the-art models on RF100-VL. We present our zero-shot and few-shot results below. See Appendix A for additional implementation details and Appendix F for semi-supervised and fully supervised results.

4.1 Metrics

Each dataset within RF100-VL is independently evaluated using AP. We report the average accuracy per super-category to simplify analysis. RF100-VL includes datasets that are out-of-distribution from typical internet-scale pre-training data, making it particularly challenging (even for VLMs). To construct the few-shot split, we follow the K-shot dataset creation process established by [49]. Importantly, all methods across data regimes are evaluated on the same fully annotated test set. In Table 1, we highlight that prior methods report different results on COCO and OdinW than our reproduced results. YOLOv8 [16] and YOLOv11 [18] achieve slightly different performance on COCO because the original results are reported using Ultralytics, whereas our results are computed using pycocotools. Importantly, this discrepancy in tooling yields a larger disparity on RF100-VL, discussed further in Appendix B. Further, we find that Qwen2.5-VL evaluates on ODinW using a referential grounding protocol (reported, see GitHub issue) instead of a traditional object detection protocol (ours). Specifically, referential grounding prompts a model with only the true positive classes in each test image, while object detectors prompt a model with *all* classes. The former dramatically reduces the number of false positives. We evaluate Gemini 2.5 Pro using both protocols for completeness.

4.2 Empirical Analysis of Results

State-of-the-Art Zero-Shot and Few-Shot Models Struggle on Roboflow100-VL. RF100-VL is a much harder dataset than prior open-vocabulary object detection benchmarks. Specifically, GroundingDINO achieves 49.2 mAP on ODinW-13, but only reaches 16 mAP on RF100-VL. Similar trends can be seen with Qwen2.5-VL and Gemini 2.5 Pro (cf. Table 1). Notably, both RF100-VL and ODinW-13 are sourced from Roboflow Universe, but our dataset is carefully curated to evaluate performance on target concepts not typically found in internet-scale pre-training.

Open-Vocabulary Object Detectors Outperform MLLMs. We find that open-vocabulary object detectors like Detic, GroundingDINO, OWLv2, and MQ-GLIP consistently outperform MLLMs like Qwen 2.5 VL, Gemini 2.5 Pro, despite these MLLMs pre-training on orders of magnitude more data. We posit that this poor performance can be attributed to MLLMs not reporting per-box confidence scores or ensuring that predictions don't overlap (e.g. non-maximal suppression). This highlights the advantage of task-specific architectures over generalist models.

Multi-Modal Annotator Instructions Provide Limited Benefit. Somewhat surprisingly, state-of-the-art MLLMs struggle to benefit from multi-modal annotator instructions. In fact, prompting with instructions provides inconsistent benefit compared to prompting with class names (e.g. Qwen2.5VL improves but Gemini 2.5 Pro degrades considerably). Intuitively, we expect annotator instructions to improve object detection performance by resolving semantic ambiguity in class names and providing rich contextual information. However, we posit that this performance decline can be attributed to the fact that MLLMs are instruction-tuned for open vocabulary detection with rigid prompt structures, making it difficult to effectively leverage additional contextual information.

Large-Scale Pre-Training Improves Fine-Tuned Few-Shot Performance in Specialists. We find that fine-tuning GroundingDINO [23] achieves the best few-shot performance, significantly outperforming all YOLO variants by more than 10%. Notably, all gradient-based fine-tuning baselines outperform in-context visual prompting and multi-modal prompting methods, suggesting that in-context prompting provides limited benefit for rare classes not seen in pre-training. We posit that GroundingDINO's large-scale task-specific pre-training makes it easier to learn new concepts during fine-tuning.

Do COCO Detectors Generalize Beyond COCO? Real-time object detectors are often optimized for COCO, assuming better performance on COCO translates to real-world improvements. However, real-world datasets (such as those in RF100-VL) are often much smaller and more diverse than COCO, challenging this assumption. Specifically, although RF100-VL has half as many images as

Table 2: **Roboflow100-VL Benchmark.** We evaluate the zero-shot, few-shot, semi-supervised, and fully-supervised performance of state-of-the-art methods on the RF100-VL benchmark. We find that RF100-VL is particularly challenging for zero-shot and few-shot approaches, with most methods struggling to achieves 10% mAP averaged over all 100 datasets. Notably, we find that GroundingDINO achieves the best zero-shot and few-shot accuracy. We use a double horizontal bar to separate specialist models from generalist MLLMs.

Method	Aerial	Document	Flora & Fauna	Industrial	Medical	Sports	Other	All
Zero-Shot								
Detic [60]	12.2	4.5	17.9	6.0	0.8	7.6	11.2	9.5
GroundingDINO [26]	21.8	7.9	28.2	10.3	2.1	13.0	18.1	15.7
OWLv2 [32] (Class Names Only)	15.3	7.3	17.7	6.3	0.8	9.7	10.2	10.1
MQ-GLIP-Text [52](Class-Names Only)	11.9	9.7	22.6	7.7	1.4	9.2	14.1	12.0
Qwen 2.5 VL (72B) [2] (Class Names Only)	4.6	3.8	10.1	3.8	1.6	5.9	5.4	5.4
Qwen 2.5 VL (72B) [2] (Instructions Only)	5.4	4.8	14.5	5.4	1.7	7.5	7.5	7.4
Gemini 2.5 Pro [9] (Class Names Only)	8.4	13.3	22.4	9.7	3.5	11.2	17.1	13.3
Gemini 2.5 Pro [9] (Instructions Only)	4.2	10.6	9.9	3.2	0.9	6.1	7.2	6.1
Few-Shot (10 shots)								
Detic [60] w/ Federated Loss [30]	19.5	19.6	28.4	25.9	8.5	26.6	25.7	22.8
MQ-GLIP-Image [52] (Images Only)	4.4	3.0	13.0	3.8	1.4	7.4	6.8	6.2
MQ-GLIP [52] (Class Names + Images)	11.9	9.2	22.6	7.7	1.4	9.3	14.1	12.0
GroundingDINO [26]	31.8	29.6	40.8	37.5	17.9	33.1	32.6	33.3
YOLOv8n [16]	13.8	21.8	21.9	23.9	11.8	13.3	18.6	20.1
YOLOv8n [16] w/ Federated Loss [30]	13.9	22.8	22.1	26.6	14.9	13.0	19.7	21.6
YOLOv8s [16]	14.3	23.4	20.8	28.1	13.7	14.3	21.0	20.7
YOLOv8s [16] w/ Federated Loss [30]	15.8	25.9	25.9	26.5	15.2	18.4	22.3	23.4
YOLOv8m [16]	16.3	20.6	23.3	23.4	13.7	20.6	20.6	21.1
YOLOv8m [16] w/ Federated Loss [30]	18.2	25.9	24.0	26.2	15.9	19.1	21.2	23.1
Qwen 2.5 VL (72B)[2] (Instructions + Images)	5.7	6.4	14.5	5.5	1.6	7.2	6.7	7.5
Gemini 2.5 Pro [9] (Images)	6.0	11.3	16.4	6.4	2.0	7.8	10.4	9.2
Gemini 2.5 Pro [9] (Instructions + Images)	5.4	13.1	15.6	6.8	2.2	6.7	10.3	9.2

COCO, it has more than seven times as many classes (cf. Fig. 5). Interestingly, we find that models that achieved higher performance on COCO did not necessarily improve real-world performance on RF100-VL. For example, YOLOv11 outperforms YOLOv8 on COCO but performs similarly to YOLOv8 across all three tested sizes (nano, small, medium) on RF100-VL. This suggests that newer YOLO models may be overfitting to COCO, as gains on that dataset don't transfer to real-world datasets. Lastly, we find that increasing model size leads to smaller performance improvements on RF100-VL compared to COCO. The performance difference between the smallest and largest models within a model family is at most 2.6 mAP, suggesting that simply increasing model capacity may not lead to significant performance gains on RF100-VL.

4.3 CVPR 2025 Foundational FSOD Chalenge

We are hosting a challenge at CVPR 2025 to encourage broad community involvement in addressing the problem of aligning foundation models to target concepts with few-shot visual examples and rich textual descriptions. To incentivize participation, top teams can win cash prizes. Importantly, we use a subset of 20 datasets from RF100-VL for this challenge to lower the barrier to entry. Although our competition will continue until June 8th 2025, we have already received submissions from seven teams (some submissions are private). We present the current top three teams in Table 3.

4.4 Limitations and Future Work

Reliance on Crowdsourced Annotations. All our datasets are sourced from Roboflow Universe, a community platform where anyone can upload dataset annotations. Although this allows us to source diverse datasets, it introduces uncertainty regarding overall annotation quality. While we manually inspect and re-annotate all datasets to ensure quality to the best of our ability, verifying annotations in specialized domains like medical imaging remains a significant challenge.

Generated Annotator Instructions May Not Reflect Real Instructions. Our annotator instructions are automatically generated by GPT-4o and are manually verified for correctness. However, they may not fully reflect the nuances of real-world instructions typically developed alongside dataset collection. We encourage the community to release real annotator instructions generated through iterative discussions between annotators and stakeholders. Furthermore, although our annotator instructions provide high-level class descriptions, they often do not directly incorporate image

Table 3: **CVPR 2025 Foundational FSOD Challenge with Roboflow20-VL**

Method	Aerial	Document	Flora & Fauna	Industrial	Medical	Sports	Other	All
Zero-Shot								
Detic [60]	4.1	1.4	22.2	6.3	0.1	1.0	9.7	8.4
GroundingDINO [26]	30.6	5.0	33.9	13.0	0.4	5.5	16.8	17.1
OWLv2 [32] (Class Names Only)	30.4	3.5	22.1	11.6	0.1	3.2	6.8	11.6
MQ-GLIP-Text [52](Class-Names Only)	29.8	2.5	31.0	5.5	0.4	6.3	10.8	13.6
Qwen 2.5 VL (72B) [2] (Class Names Only)	3.6	3.5	9.7	2.7	0.0	9.5	3.8	5.0
Qwen 2.5 VL (72B) [2] (Instructions Only)	4.7	7.7	13.2	5.0	0.4	11.5	5.6	7.2
Gemini 2.5 Pro [9] (Class Names Only)	9.6	11.2	27.2	14.9	1.9	14.1	16.6	15.5
Gemini 2.5 Pro [9] (Instructions Only)	2.7	13.8	12.6	5.5	0.0	8.0	4.1	6.8
Few-Shot (10 shots)								
Detic w/ Federated Loss [30]	11.6	14.3	30.8	24.7	8.9	17.4	21.0	20.3
GroundingDINO [26]	39.8	34.5	45.6	37.8	23.3	26.3	24.7	33.3
MQ-GLIP-Image [52] (Images Only)	1.7	1.1	18.8	1.8	0.1	6.8	7.2	6.8
MQ-GLIP [52] (Class Names + Images)	29.8	2.5	31.1	5.5	0.4	6.3	11.0	13.7
Qwen 2.5 VL (72B) [2] (Instructions + Images)	4.9	9.2	15.1	2.8	0.2	8.5	5.4	7.1
Gemini 2.5 Pro [9] (Images Only)	11.1	16.3	24.8	4.5	0.2	12.8	5.1	10.9
Gemini 2.5 Pro [9] (Instructions + Images)	9.8	16.2	24.8	4.5	0.2	12.8	4.8	10.7
Challenge Submissions								
NJUST KMG	42.8	26.2	46.9	29.0	22.3	24.1	34.0	33.8
Fighting Mongooses	32.9	31.1	39.4	35.0	23.8	21.1	24.6	30.2
Guardians of Ga'hoole	33.6	23.3	18.1	22.0	2.2	8.7	19.1	18.5

evidence to identify typical cases, edge cases, and negative examples. Future work should explore how to create better automatic annotator instructions.

Generalist and Specialist Models have Complementary Strengths. Although specialist models like GroundingDINO [26] outperform generalist models like Qwen2.5-VL [2], MLLMs can more easily process few-shot visual examples and rich textual descriptions. Future work should combine the versatility of MLLMs with the precision of specialist models.

5 Conclusion

In this paper, we introduce Roboflow100-VL, a large-scale benchmark to evaluate state-of-the-art VLMs on concepts not typically found in internet-scale pre-training. RF100-VL is curated to evaluate detection performance on out-of-distribution tasks (e.g. material property estimation, defect detection, and contextual action recognition) and imaging modalities (e.g. X-rays, thermal spectrum data, and aerial imagery) using a few visual examples and rich textual descriptions. We find that state-of-the-art models struggle on this challenging benchmark, demonstrating the limitations of existing methods, highlighting opportunities to develop better algorithms that effectively use multi-modal annotator instructions. We hope that RF100-VL will be a rigorous test-bench for future VLMs and MLLMs.

6 Acknowledgments

This work was supported in part by compute provided by NVIDIA, and the NSF GRFP (Grant No. DGE2140739).

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. "Qwen2. 5-vl technical report". In: *arXiv preprint arXiv:2502.13923* (2025).

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. "nuscenes: A multimodal dataset for autonomous driving". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[4] Nadine Chang, Francesco Ferroni, Michael J Tarr, Martial Hebert, and Deva Ramanan. "Thinking Like an Annotator: Generation of Dataset Labeling Instructions". In: *arXiv preprint arXiv:2306.14035* (2023).

[5] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. "Are we on the right way for evaluating large vision-language models?" In: *arXiv preprint arXiv:2403.20330* (2024).

[6] Qiang Chen, Xiangbo Su, Xinyu Zhang, Jian Wang, Jiahui Chen, Yunpeng Shen, Chuchu Han, Ziliang Chen, Weixiang Xu, Fanrong Li, et al. "LW-DETR: a transformer replacement to yolo for real-time detection". In: *arXiv preprint arXiv:2406.03459* (2024).

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[8] Floriana Ciaglia, Francesco Saverio Zuppichini, Paul Guerrie, Mark McQuade, and Jacob Solawetz. "Roboflow 100: A rich, multi-domain object detection benchmark". In: *arXiv preprint arXiv:2211.13523* (2022).

[9] Google DeepMind. *Introducing Gemini 2.0: our new AI model for the agentic era*. Dec. 2024. URL: `https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/`.

[10] Yu Du, Fangyun Wei, Zihe Zhang, Miaojing Shi, Yue Gao, and Guoqi Li. "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*. 2022, pp. 14084–14093.

[11] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. "Clip-adapter: Better vision-language models with feature adapters". In: *International Journal of Computer Vision* 132.2 (2024), pp. 581–595.

[12] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. "Open-vocabulary object detection via vision and language knowledge distillation". In: *arXiv preprint arXiv:2104.13921* (2021).

[13] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. "Open-vocabulary object detection via vision and language knowledge distillation". In: *arXiv preprint arXiv:2104.13921* (2021).

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. "Masked autoencoders are scalable vision learners". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16000–16009.

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "Lora: Low-rank adaptation of large language models". In: *arXiv preprint arXiv:2106.09685* (2021).

[16] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. *Ultralytics YOLO*. Version 8.0.0. Jan. 2023. URL: `https://github.com/ultralytics/ultralytics`.

[17] Gaoussou Youssouf Kebe, Padraig Higgins, Patrick Jenkins, Kasra Darvish, Rishabh Sachdeva, Ryan Barron, John Winder, Donald Engel, Edward Raff, Francis Ferraro, and Cynthia Matuszek. "A Spoken Language Dataset of Descriptions for Speech-Based Grounded Language Learning". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. 2021.

[18] Rahima Khanam and Muhammad Hussain. "Yolov11: An overview of the key architectural enhancements". In: *arXiv preprint arXiv:2410.17725* (2024).

[19] Mehar Khurana, Neehar Peri, Deva Ramanan, and James Hays. "Shelf-Supervised Multi-Modal Pre-Training for 3D Object Detection". In: *arXiv preprint arXiv:2406.10115* (2024).

[20] Weicheng Kuo, Yin Cui, Xiuye Gu, AJ Piergiovanni, and Anelia Angelova. "F-vlm: Open-vocabulary object detection upon frozen vision and language models". In: *arXiv preprint arXiv:2209.15639* (2022).

[21] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. "Seed-bench: Benchmarking multimodal llms with generative comprehension". In: *arXiv preprint arXiv:2307.16125* (2023).

[22] Chunyuan Li, Haotian Liu, Liunian Harold Li, Pengchuan Zhang, Jyoti Aneja, Jianwei Yang, Ping Jin, Houdong Hu, Zicheng Liu, Yong Jae Lee, and Jianfeng Gao. "ELEVATER: A Benchmark and Toolkit for Evaluating Language-Augmented Visual Models". In: *Neural Information Processing Systems* (2022).

[23] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. "Grounded language-image pre-training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10965–10975.

[24] Zijing Liang, Yanjie Xu, Yifan Hong, Penghui Shang, Qi Wang, Qiang Fu, and Ke Liu. "A Survey of Multimodel Large Language Models". In: *Proceedings of the 3rd International Conference on Computer, Artificial Intelligence and Control Engineering*. 2024, pp. 405–409.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context". In: *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*. Springer. 2014, pp. 740–755.

[26] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection". In: *arXiv preprint arXiv:2303.05499* (2023).

[27] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. "Mmbench: Is your multi-modal model an all-around player?" In: *European conference on computer vision*. Springer. 2024, pp. 216–233.

[28] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. "Learn to explain: Multimodal reasoning via thought chains for science question answering". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 2507–2521.

[29] Yechi Ma, Neehar Peri, Shuoquan Wei, Wei Hua, Deva Ramanan, Yanan Li, and Shu Kong. "Long-Tailed 3D Detection via 2D Late Fusion". In: *arXiv preprint arXiv:2312.10986* (2023).

[30] Anish Madan, Neehar Peri, Shu Kong, and Deva Ramanan. "Revisiting few-shot object detection with vision-language models". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 19547–19560.

[31] Sachit Menon and Carl Vondrick. "Visual Classification via Description from Large Language Models". In: *The Eleventh International Conference on Learning Representations (ICLR)*. 2023.

[32] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. "Scaling Open-Vocabulary Object Detection". In: *arXiv preprint arXiv:2306.09683* (2023).

[33] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. "Simple open-vocabulary object detection". In: *European Conference on Computer Vision*. Springer. 2022, pp. 728–755.

[34] Aljosa Osep, Tim Meinhardt, Francesco Ferroni, Neehar Peri, Deva Ramanan, and Laura Leal-Taixe. "Better Call SAL: Towards Learning to Segment Anything in Lidar". In: *ECCV*. 2024.

[35] Hongpeng Pan, Shifeng Yi, Shouwei Yang, Lei Qi, Bing Hu, Yi Xu, and Yang Yang. "The Solution for CVPR2024 Foundational Few-Shot Object Detection Challenge". In: *arXiv preprint arXiv:2406.12225* (2024).

[36] Shubham Parashar, Zhiqiu Lin, Tian Liu, Xiangjue Dong, Yanan Li, Deva Ramanan, James Caverlee, and Shu Kong. "The Neglected Tails in Vision-Language Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 12988–12997.

[37] Neehar Peri, Achal Dave, Deva Ramanan, and Shu Kong. "Towards Long-Tailed 3D Detection". In: 2023.

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

[39] Roboflow. *Roboflow Inference*. URL: `https://github.com/roboflow/inference`.

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115 (2015), pp. 211–252.

[41] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. "Laion-5b: An open large-scale dataset for training next generation image-text models". In: *Advances in neural information processing systems* 35 (2022), pp. 25278–25294.

[42] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. "Objects365: A Large-Scale, High-Quality Dataset for Object Detection". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 8429–8438. DOI: `10.1109/ICCV.2019.00852`.

[43] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. "Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 2556–2565.

[44] Kihyuk Sohn, Zizhao Zhang, Chun-Liang Li, Han Zhang, Chen-Yu Lee, and Tomas Pfister. "A simple semi-supervised learning framework for object detection". In: *arXiv preprint arXiv:2005.04757* (2020).

[45] Ayca Takmaz, Cristiano Saltori, Neehar Peri, Tim Meinhardt, Riccardo de Lutio, Laura Leal-Taixe, and Aljosa Osep. "Towards Learning to Complete Anything in Lidar". In: *International Conference on Machine Learning (ICML)*. 2025.

[46] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. "Winoground: Probing vision and language models for visio-linguistic compositionality". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5238–5248.

[47] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. "Eyes wide shut? exploring the visual shortcomings of multimodal llms". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 9568–9578.

[48] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. "A comprehensive survey of continual learning: Theory, method and application". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[49] Xin Wang, Thomas E. Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. "Frustratingly Simple Few-Shot Object Detection". In: *International Conference on Machine Learning (ICML)*. 2020.

[50] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. "Growing a brain: Fine-tuning by increasing model capacity". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2471–2480.

[51] Wenhao Wu, Zhun Sun, and Wanli Ouyang. "Revisiting classifier: Transferring vision-language models for video recognition". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 37. 3. 2023, pp. 2847–2855.

[52] Yifan Xu, Mengdan Zhang, Chaoyou Fu, Peixian Chen, Xiaoshan Yang, Ke Li, and Changsheng Xu. "Multi-modal queried object detection in the wild". In: *Advances in Neural Information Processing Systems* 36 (2024).

[53] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. "Meta r-cnn: Towards general solver for instance-level low-shot learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9577–9586.

[54] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. "Modeling context in referring expressions". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer. 2016, pp. 69–85.

[55] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. "Mm-vet: Evaluating large multimodal models for integrated capabilities". In: *arXiv preprint arXiv:2308.02490* (2023).

[56] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. "Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 9556–9567.

[57] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. "Adding conditional control to text-to-image diffusion models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 3836–3847.

[58] Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. "Tip-adapter: Training-free clip-adapter for better vision-language modeling". In: *arXiv preprint arXiv:2111.03930* (2021).

[59] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, et al. "Regionclip: Region-based language-image pretraining". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16793–16803.

[60] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. "Detecting twenty-thousand classes using image-level supervision". In: *European Conference on Computer Vision*. Springer. 2022, pp. 350–368.

A Implementation Details

We present additional implementation details to reproduce our baseline experiments below. Our code is available on GitHub.

Detic. We use Detic [60] with a SWIN-L backbone for all zero-shot experiments. Additionally, we use the model checkpoint trained on LVIS, COCO and ImageNet-21K. We use class names provided as text prompts for Detic's CLIP classifier.

GroundingDINO. We use GroundingDINO [26] with pretrained weights from mmdetection (MM-GroundingDINO-L*). We prompt the model with all the class names combined into a single prompt. We fine-tune GroundingDINO on each few-shot dataset for 5 epochs with a batch size 4 and learning rate of 3e-4.

MQ-GLIP. MQ-Det [52] 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. Laslty, we use class names as the text prompts and few-shot visual examples as visual prompts.

OWLv2. We use OWLv2 [32] as implemented in the Roboflow inference package [39]. We prompt the model with each class name independently and post-process predictions with non-maximal suppression. We release the benchmarking code adapted for RF100-VL on GitHub.

Qwen-2.5VL. We conduct all experiments using the "qwen2.5-vl-72b-instruct" model via API. We prompt the model based on guidelines from Qwen's official documentation. We also improve the 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: {class names} in the image and output the coordinates in JSON format."

Single-Class Detection Prompt

"Locate every {class name} in the image and output the coordinates in JSON format."

Gemini 2.5 Pro. We conduct all experiments using the Gemini API with the "gemini-2.5-pro-preview-03-25" model. We prompt the model based on guidelines from Gemini's official documentation, but also improve the base prompt through small-scale validation on multiple datasets and select the best prompt:

System Prompt

"Return bounding boxes as a JSON array with labels. Never return masks or code fencing."

Multi-Class Detection Prompt

"Detect the 2d bounding boxes of the following objects: {class names}"

Single-Class Detection Prompt

"Detect all 2d bounding boxes of {class name}."

Prompting with Rich Textual Descriptions To evaluate Qwen and Gemini with dataset-specific annotator instructions, we appended the following prompt after our main prompt:

"Use the following annotator instructions to improve detection accuracy: {annotator 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 (extracted using GPT-4o) when using the single-class detection prompt.

Prompting with Few-Shot Visual Examples We provide one image at a time to Qwen and Gemini to mimic their turn-based pre-training. We use all few-shot images when prompting Gemini. However, we only use three images when prompting Qwen due to API limitations.

We prompt Gemini with native resolution images, but limit Qwen's few-shot visual examples to a minimum of 4*28*28 pixels and a maximum of 12800*28*28 pixels due to API limitations. To manage costs, we limit Gemini to only output 8192 tokens per request. We do not set any token limits for Qwen. Lastly, we implement a robust parser to handle minor JSON formatting errors. In some cases with many few-shot image examples, the API fails to return a valid response for requests of excessive size. In such cases, we simply assign a score of 0 AP for those images. Due to Gemini and Qwen not always predicting a confidence score for their bounding boxes, we set it to 1.0 by default.

YOLOv8 and YOLOv11. We train our YOLOv8 [16] and YOLOv11 [18] family of models using the Ultralytics package with default parameters. For all models, we follow the established protocol in Ciaglia et. al. [8] and train for 100 epochs with a batch size of 16. However, we evaluate all YOLO models using pycocotools instead of Ultralytics (cf. Appendix B)

B Additional Evaluation Details

We find that metrics reported with pycocotools (500 maxDets) differs significantly from those reported by Ultralytics on RF100-VL (cf. Table 4). Notably, all YOLO models report metrics using Ultralytics' implementation of mAP by default. Our preliminary investigation, supported by similar observations on Github, suggest that this disparity can be largely attributed to differences in the integration method of the precision-recall curve. Ultralytics uses a trapezoidal sum, which inflates model performance by as much 3.4% compared to pycocotools. We choose to report results for YOLO models using pycocotools in the main paper to standardize our results with our other baselines.

Table 4: **Impact of Evaluation Toolkit on RF100-VL Performance.** We find that the Ultralytics mAP calculation significantly over-estimates mAP compared with pycocotools. For fair comparison with other baselines, we choose to report metrics using pycocotools.

Method	pycocotools mAP (Ours)	Ultralytics mAP
YOLOv8n [16]	54.9	57.4
YOLOv11n [18]	55.3	57.3
YOLOv8s [16]	56.2	59.2
YOLOv11s [18]	56.2	59.0
YOLOv8m [16]	56.4	59.8
YOLOv11m [18]	56.5	59.7

C Ablation on Prompting MLLMs

We evaluate Gemini 2.5 Pro and Qwen 2.5-VL performance on RF100-VL using two prompting strategies: single-class prompting and multi-class prompting. The single-class prompting strategy separately performs a forward pass for each class and merges the results per image. The multi-class prompting strategy performs a single forward pass for all classes. Both Gemini 2.5 Pro and Qwen

2.5-VL recommend the single-class prompting strategy. Importantly, we do not perform non-maximal suppression for either strategy as both MLLMs do not report confidence scores per box.

Interestingly, we observe that Qwen2.5VL performs better with single-class prompts, while Gemini performs better with multi-class prompts. We posit that this can be attributed to Qwen's extensive referential object detection pre-training, which typically requires detecting a single class. In contrast, Gemini achieves better performance with multi-class prompting, which is more aligned with traditional object detection setups. We argue that multi-class prompting should be the default for assessing a MLLM's object detection capabilities since this more closely mirrors standard object detection protocols.

Table 5: **Analysis of Prompting Strategy.** MLLMs typically evaluate detection performance with single-class prompts. We find that Qwen2.5VL achieves better performance with single-class prompts, while Gemini 2.5 Pro achieves better performance with multi-class prompts. We advocate for multi-class prompting since this more closely matches object detection evaluation.

Method	Single-Class Prompt	Multi-Class Prompt
Gemini 2.5 Pro	8.6	13.3
Qwen 2.5-VL (72B)	7.6	5.4

D Comparing Different Model Sizes

In Figure 6, we evaluate the performance of the Gemini model family over time (e.g. Gemini Flash 2.0 was released before Gemini Flash 2.5). Although Gemini has not been explicitly fine-tuned on RF100-VL, we see a significant increase in performance. This suggests that Gemini is making real progress towards zero-shot open-vocabulary object detection in the wild. Unsurprisingly, base MLLMs outperform faster distilled models (e.g. Gemini 2.5 Pro achieves 2.4% better performance than Gemini Flash 2.5), but distilled models provide considerably better performance per dollar. Importantly, all models are prompted with multi-class prompts (cf. Appendix C).

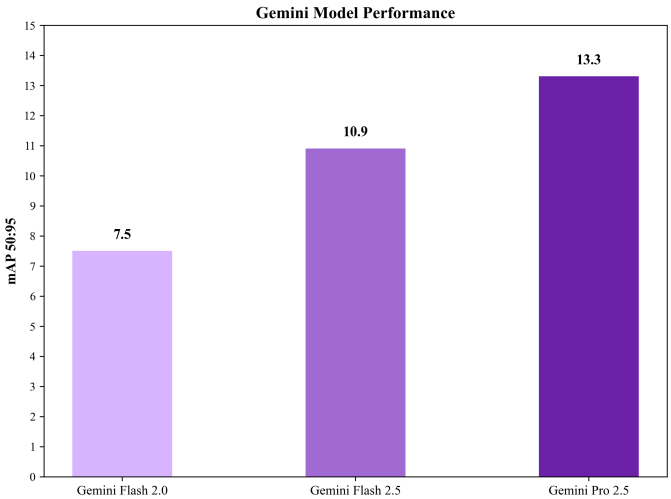


Figure 6: **Gemini Improves on RF100-VL over Time**. Despite not explicitly fine-tuning on RF100-VL, we find that newer Gemini models consistently improve over older models on our benchmark. This suggests that Gemini is making real progress towards improving zero-shot open-vocabulary detection in-the-wild.

E Ablation on Few-Shot Split Selection

Prior work typically selects few-shot training examples at random. However, Madan et. al. [30] demonstrates that the specific few-shot examples used for fine-tuning greatly affects target class performance. Specifically, Madan et. al. selects the most informative K-shot examples for each class in nuImages [3] by evaluating Detic w/ Federated Fine-Tuning's class-wise performance on a held-out validation set. For instance, in a 5-shot task with three random splits, we may select our five-shot car examples from split 1, our five-shot bicycles from split 3, and our five-shot debris from

split 2 based on which split has the highest per-class accuracy. As shown in Table 6, this "best split [30]" approach consistently outperforms random selection.

Despite the effectiveness of this approach, it has two primary limitations. First, it uses the validation performance of a specific model to inform few-shot selection. This inherently biases the few-shot images towards a particular model. Next, Madan et. al.'s proposed algorithm is computationally expensive since it requires fine-tuning a model on many candidate few-shot splits. This approach is computationally infeasible with RF100-VL's 100 datasets.

To address these two issues, we propose a learning-free approach that leverages the key insight from Madan et. al's analysis: the "best" examples are typically large and unoccluded. Concretely, we generate K random candidate few-shot splits for each class and pick the split that has the largest average bounding box area. Similar to Madan et. al., in a 5-shot task with three random splits, we may select our five-shot car examples from split 1, our five-shot bicycles from split 3, and our five-shot debris from split 2. We evaluate our proposed sampling strategy on nuImages and find that this approach performs better than random, but underperforms Madan et. al.'s approach. Future work should consider more effective strategies for selecting the "best" few-shot examples for concept alignment.

Table 6: **"Best" Split Construction**. We evaluate the quality of few-shot example selection using a (1) random baseline, (2) Madan et al.'s "best split" approach, which chooses per-class few-shot examples based on Detic w/ Federated Fine-Tuning's validation accuracy, and (3) our proposed learning-free method that selects splits with the largest average bounding box area. While Madan et al.'s method performs the best, it is biased towards Detic and is computationally expensive. Our approach offers a tractable alternative that improves over the random baseline.

Approach	Average Precision (AP)			
	`All`	`Many`	`Medium`	`Few`
Detic (Zero-Shot) [60]	14.40	25.83	16.59	2.32
Detic w/ Federated Fine-Tuning *(5-shots, Random Split)*	16.58	27.12	19.71	4.13
Detic w/ Federated Fine-Tuning *(5-shots, Best Split [30])*	**18.30**	**28.66**	**21.81**	**5.56**
Detic w/ Federated Fine-Tuning *(5-shots, Best Split, Ours)*	16.94	28.41	20.32	3.45
Detic w/ Federated Fine-Tuning *(10-shots, Random Split)*	17.24	28.07	20.71	4.18
Detic w/ Federated Fine-Tuning *(10-shots, Best Split [30])*	**18.24**	**28.63**	22.00	**5.19**
Detic w/ Federated Fine-Tuning *(10-shots, Best Split, Ours)*	17.48	26.36	**22.42**	4.32

F Semi-Supervised and Fully Supervised Results

We present results from semi-supervised and fully-supervised baselines in Table 7. Importantly, these models are evaluated on the same data splits as our zero-shot and few-shot baselines. To construct the semi-supervised split, we randomly sample 10% of the training set.

Semi-Supervised Baselines. We evaluate variants of YOLO [16, 18] and YOLO with STAC [44] trained on 10% of each dataset in RF100-VL. STAC generates high-confidence pseudo-labels for localized objects in unlabeled images and updates the model by enforcing consistency through strong augmentations. We follow the training protocol defined by Sohn et. al. [44]. First, we train a teacher model on the labeled subset of the data. Then, we use the teacher model to pseudo-label the remaining unlabeled subset of the data. We keep all detections above a confidence C, where the confidence tuned to maximize the F1 score of the teacher model on a validation set. Finally, we combine the subset of data with true ground truth labels and the subset with pseudo-labels to form a training set for a student model of the same architecture. We train this student model until convergence with heavy augmentations. We use the same hyperparameters as our supervised YOLOv8 and YOLOv11 implementation. Because YOLO models already train with significant augmentation, we don't add any new augmentations for the student training.

Fully-Supervised Baselines. We benchmark YOLOv8 [16], YOLOv11 [18], and LW-DETR [6] on all datasets within RF100-VL. YOLOv8, developed by Ultralytics, builds on the YOLOv5 architecture with improvements in model scaling and architectural refinements. YOLOv11 adds more architecture improvements, and is primarily validated on COCO. LW-DETR is a lightweight detection transformer that outperforms YOLO models for real-time object detection, and is SOTA on the original Roboflow100 [8] dataset, the predecessor to RF100-VL. Its architecture consists of a ViT encoder, a projector, and a shallow DETR decoder. This baseline serves as an upper bound on

performance, though in rare cases, few-shot foundation models may surpass it when the target dataset only has a few examples.

Semi-Supervised Learners are Data Efficient. We find that leveraging simple semi-supervised learning algorithms like STAC [44] significantly improves model performance when learning with limited labels. In half (7 out of 14) of combinations of model size and data domain, semi-supervised learners improved mAP at least as much as stepping up a model size. For example, YOLOv8s (small) trained on 10% labeled data (and 90% STAC psuedo-labels) achieves better performance overall than YOLOv8m (medium) trained on just 10% labeled data.

Table 7: **Roboflow100-VL Semi-Supervised and Fully-Supervised Benchmark.** We find that semi-supervised learners are able to reach nearly 80% of the performance of fully supervised models using 10% labeled data.

Method	Aerial	Document	Flora & Fauna	Industrial	Medical	Sports	Other	All
Semi-Supervised (10% Labels)								
YOLOv8n [16]	32.7	34.3	41.1	50.4	28.7	29.1	37.9	38.7
YOLOv8n [16] w/ STAC [44]	34.4	37.8	42.3	51.7	30.6	33.0	40.6	40.7
YOLOv8s [16]	36.4	39.0	41.5	51.7	31.1	38.0	40.9	41.3
YOLOV8s [16] w/ STAC [44]	37.4	39.9	41.5	52.0	32.5	39.9	43.4	42.5
YOLOv8m [16]	36.8	42.3	40.0	51.9	31.5	40.8	41.7	41.7
YOLOv8m [16] w/ STAC [44]	38.2	41.7	42.1	52.5	32.1	42.4	44.0	42.9
Fully-Supervised								
YOLOv8n [16]	49.5	56.4	53.3	64.0	49.2	48.8	54.3	54.9
YOLOv11n [18]	50.8	56.0	54.3	64.3	49.0	50.0	54.4	55.3
YOLOv8s [16]	51.8	58.0	54.7	64.7	49.2	51.2	55.9	56.2
YOLOv11s [18]	51.7	58.1	54.4	64.7	49.2	51.8	56.6	56.2
LW-DETRs [6]	52.7	59.3	55.1	68.2	51.4	54.9	57.7	58.0
YOLOv8m [16]	52.4	59.2	54.3	64.9	48.5	52.5	56.6	56.4
YOLOv11m [18]	52.1	59.6	54.3	65.3	48.7	52.4	56.8	56.5
LW-DETRm [6]	55.2	59.1	57.0	68.5	52.5	56.5	59.9	59.4

G Analysis of Accuracy vs. Parameter Count

In Figure 7, we observe a counter-intuitive trend: larger models perform worse in our evaluations. This is likely due to the mismatch between general-purpose MMLMs and specialized object detectors. Despite being the largest model pre-trained on the most data, Qwen2.5-VL (72B) underperforms GroundingDINO in the zero-shot setting and is also considerably slower. Interstingly, we find that GroundingDINO fine-tuned on few-shot examples surpasses all YOLO models fine-tuned on few-shot examples, indicating that large pre-trained backbones enable more efficient fine-tuning in specialist models.

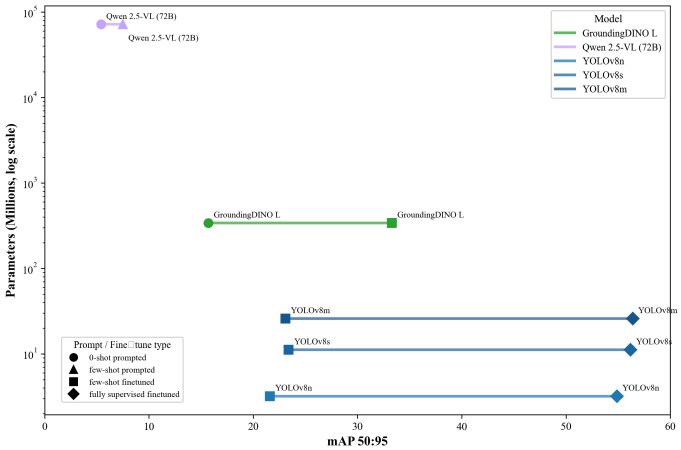


Figure 7: **Accuracy vs. Parameter Count.** Somewhat counterintuitively, we find that the model with the most parameters (Qwen2.5-VL 72B) performs worse than significantly smaller models pre-trained on less data (GroundingDINO) in the zero-shot setting. This suggests that generalist MLLMs are parameter inefficient for specialized tasks.

H Correlation Between Model Type and Per-Dataset Performance

Figure 8 presents four scatterplots comparing mAP 50:95 across different model pairs on RF100-VL, with each axis representing one model's mAP and each point labeled by a dataset index (sorted

alphabetically). These plots help identify whether certain datasets are universally easy, medium, or hard across models.

We compare Gemini vs. GroundingDINO, Qwen vs. GroundingDINO, Gemini vs. Qwen, and GroundingDINO vs. YOLO. Gemini and Qwen, as well as GroundingDINO and YOLO, show stronger linear correlations in their per-dataset scores, suggesting alignment in perceived difficulty. In contrast, comparisons between generalists (Gemini and Qwen) and specialists (GroundingDINO and YOLO) show weaker correlation. This suggests that large-scale MLLMs, likely trained on similar web data, align more closely with each other, while specialist models like GroundingDINO and YOLO show stronger consistency. These results imply that dataset difficulty levels (easy, medium, hard) may not generalize across model classes, but may be better defined within model types.

Additionally, among the top 15 datasets where Gemini outperforms Qwen and GroundingDINO, seven overlap. This suggests that Gemini may excel on datasets similar to those found in its pretraining, but struggles to generalize to novel domains.

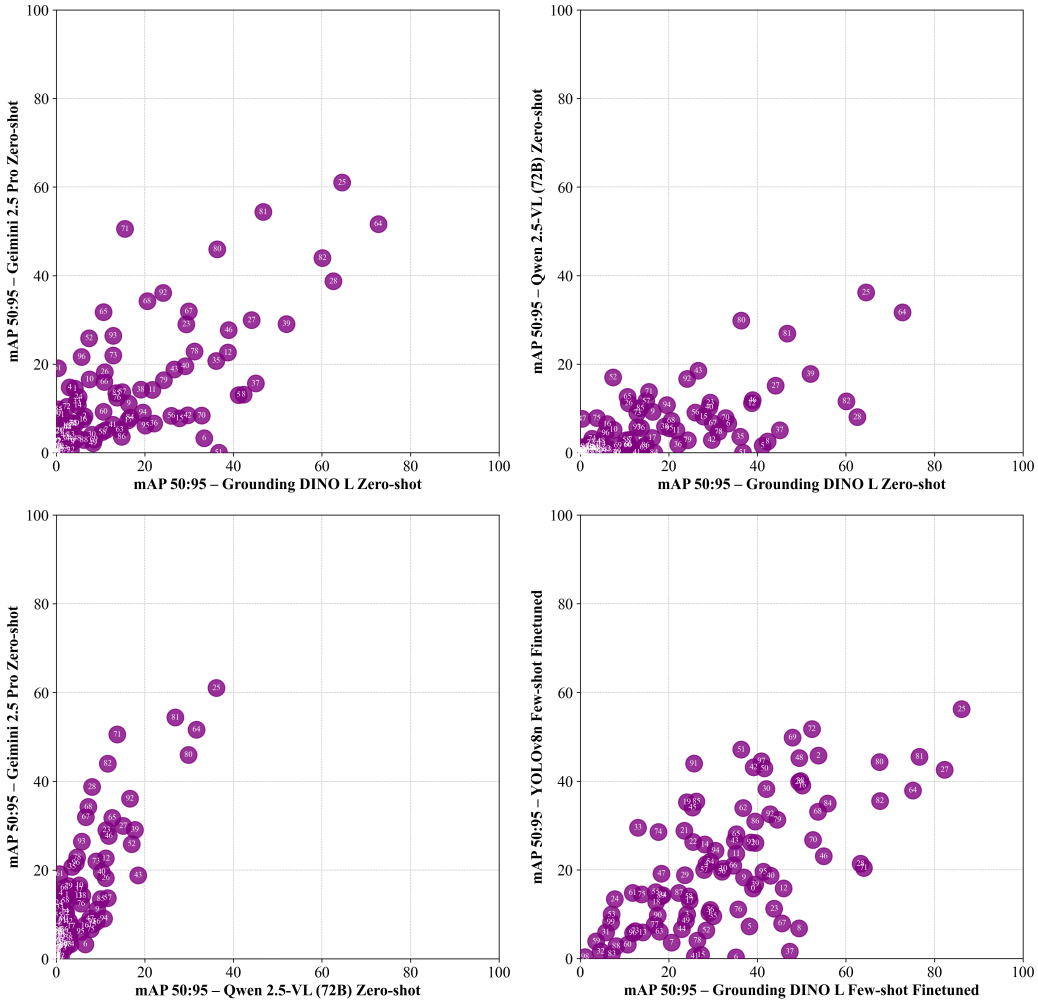


Figure 8: **Correlation Between Models Type and Performance.** We see stronger linear trends between Gemini and Qwen, and between GroundingDINO and YOLO, indicating aligned perceptions of dataset difficulty within model groups.

I Performance Variance for Few-Shot Models

Tables 8 and 9 measure the variance of YOLOv8 on RF100-VL. We use this model as a proxy for understanding few-shot learning variance and the statistical significance of our results. We train

YOLOv8n and YOLOv8s [16] with federated loss [30] ten times on each dataset, using ten different random seeds to determine model initialization and augmentation selection. We report the mean and standard deviation in two ways. In Table 8, we take the average mAP across all datasets in a given category (e.g. Industrial, Sports, All, etc.), and report the mean mAP and standard deviation across ten different runs. In Table 9, we measure the mean and standard deviation for each dataset across 10 different runs, and then report the average mean and standard deviation over each category. This will result in a higher standard deviation. Table 9 conveys the variance of a single dataset in RF100-VL and motivates averaging mAP across multiple datasets as a more stable metric.

Table 8: **Roboflow100-VL Overall Variance.** We evaluate the mean mAP and standard deviation of ten runs of YOLOv8 [16] with federated loss [30] over different subsets of Roboflow100-VL. These results can be used as a proxy to calculate whether a new entry to Table 2 is statistically significant. Unsurprisingly, averaging over 100 datasets yields a less noisy estimate of model performance

Method	Aerial	Document	Flora & Fauna	Industrial	Medical	Sports	Other	All
YOLOv8n [16]	13.5 ± .686	24.1 ± .986	21.9 ± .941	25.2 ± .446	14.3 ± .825	14.4 ± .860	20.4 ± .772	21.2 ± .201
YOLOv8s [16]	16.5 ± .731	25.2 ± 1.22	24.6 ± .471	26.4 ± .496	16.0 ± .717	19.2 ± .734	22.4 ± .901	23.2 ± .319

Table 9: **Roboflow100-VL Dataset Variance.** We evaluate the mean mAP and standard deviation over 10 runs of YOLOv8 [16] with federated loss [30] for each of the 100 datasets in Roboflow100-VL. This result helps quantify how much a model should improve on a single dataset to be statistically significant. This approach for quantifying statistical significance shows a much higher variance.

Method	Aerial	Document	Flora & Fauna	Industrial	Medical	Sports	Other	All
YOLOv8n [16]	13.5 ± 1.78	24.1 ± 2.40	21.9 ± 2.94	25.2 ± 2.28	14.3 ± 1.97	14.4 ± 2.13	20.4 ± 1.88	21.2 ± 2.29
YOLOv8s [16]	16.5 ± 1.49	24.2 ± 2.52	24.6 ± 2.88	26.4 ± 2.32	16.0 ± 2.25	19.2 ± 2.08	22.4 ± 2.38	23.2 ± 2.36

J Annotation Generation Instructions

We present our prompt for generating multi-modal annotator with GPT-4o below.

```
Pay attention to the following example annotation instructions for nu-images,
an object detection dataset:

{nuImages Annotator Instructions}

That was an example of object detection annotation instructions.

Using the above instructions as rough inspiration, come up with annotation
instructions for a dataset.

The annotation instructions should be in markdown format, and follow the
following outline:

‘‘‘markdown
# Overview
Table of contents

# Introduction
Introduction to the dataset. Introduce what task the dataset is trying to
solve. List all of the classes and provide a brief description of each class.

# Object Classes
## Class 1
### Description
Provide a description of the class, paying attention to visually distinctive
elements of the class.
### Instructions
Provide detailed instructions for how to annotate this class. Give specific
references to the class, and pay attention to the example labeled images that
```

```
will be provided. Provide specific descriptions of what not to label, if applicable.
...

## Class 2
...

## Class n
...
‘‘‘

Please pay specific attention to the provided visual example images and ground
your response in those examples. Be brief and concise, but comprehensive.
Make sure ### Instructions in each class provides visual descriptions of what
exactly to annotate.

Visual descriptions should make specific reference to how the object looks in
each image. If the object is not something everyone knows, describe its distinctive
shape, color, texture, etc. Look at the example pictures when coming up with these
instructions.

Respond with only the markdown content, no other text (and no backticks). Do not
describe the color of the bounding box, just describe how to find the spatial extent
of the object in the image.

The final markdown file should not make specific reference to the provided example
images. Those are simply to help you come up with the instructions. An annotator
should be able to recreate the annotations in the example images using your
generated instructions.

If the classes are similar, make sure the instructions specify how to disambiguate
between them (visually, which specific visual features to look for).

The visual content of the image should be used to clarify the description of each
class. Feel free to generalize about what is present in the dataset from the example
images.

Here is general metadata about the dataset:
{Metadata}

Here are the class names:
{Class Names}

Here are the example images:
{Few-Shot Example Images}
```

K Sample Annotation Instructions

We present sample annotator instructions below. We use dataset metadata, class names and few-shot visual examples and prompt GPT-4o [1] to generate annotator instructions (cf. Appendix J). We then manually verify that the instructions accurately describe the few-shot examples. These annotator instructions are from recode-waste-czvmg-fsod-yxsw.

```
# Overview
- [Introduction](#introduction)
- [Object Classes](#object-classes)
  - [Aggregate](#aggregate)
  - [Cardboard](#cardboard)
  - [Hard Plastic](#hard-plastic)
```

```
  - [Metal](#metal)
  - [Soft Plastic](#soft-plastic)
  - [Timber](#timber)

# Introduction
This dataset is designed for waste classification within different material
classes. The goal is to accurately identify and annotate different types of
waste materials for sorting and recycling purposes. The classes represented
are: Aggregate, Cardboard, Hard Plastic, Metal, Soft Plastic, and Timber.

# Object Classes

## Aggregate
### Description
Aggregate refers to small, granular materials, often irregular in shape with
rough surfaces. They generally appear as pieces of stone or concrete.

### Instructions
Annotate all visible portions of aggregate items. Ensure to include entire
objects even if occluded by other materials, estimating boundaries if necessary.
Exclude dust or very fine particles that do not form distinct objects.

## Cardboard
### Description
Cardboard objects are typically flat and have a layered texture. They may appear
as boxes or sheets.

### Instructions
Annotate only distinguishable pieces of cardboard, focusing on their flat surfaces
and any visible layering. Do not annotate cardboard that is part of another object
or soiled beyond recognition.

## Hard Plastic
### Description
Hard plastics are rigid and maintain their shape. They can be cylindrical, tubular,
or robust objects often found in industrial contexts.

### Instructions
Annotate the entire visible area of hard plastic objects, ensuring to capture their
solid structure. Avoid labeling small, indistinct pieces or any plastic that appears
flexible.

## Metal
### Description
Metal objects are robust, often shiny or reflective. They can appear as rods, sheets,
or other distinct shapes.

### Instructions
Label all distinct metal parts, taking care to capture their complete form. Avoid
labeling rust marks or indistinct metallic fragments lacking shape.

## Soft Plastic
### Description
Soft plastics are flexible and often transparent or translucent. They may appear in
the form of bags or wrappers.

### Instructions
Focus on full pieces of soft plastic material, ensuring to include areas with visible
creases or folds indicating flexibility. Do not label pieces smaller than a
```

```
recognizable package or those mixed with other materials.

## Timber
### Description
Timber objects are wooden, either rough or smooth, often elongated or rectangular.

### Instructions
Annotate the entire visible portion of timber, focusing on the grain or wood texture.
Do not label splinters or fragments that do not exhibit a clear wooden structure.
```

L Roboflow100-VL Datasets

We present a table with links to all datasets within Roboflow100-VL (fully-supervised and FSOD datasets) below.

Flora & Fauna	**Link**
aquarium-combined	FSOD, Fully Supervised
bees	FSOD, Fully Supervised
deepfruits	FSOD, Fully Supervised
exploratorium-daphnia	FSOD, Fully Supervised
grapes-5	FSOD, Fully Supervised
grass-weeds	FSOD, Fully Supervised
gwhd2021	FSOD, Fully Supervised
into-the-vale	FSOD, Fully Supervised
jellyfish	FSOD, Fully Supervised
marine-sharks	FSOD, Fully Supervised
orgharvest	FSOD, Fully Supervised
peixos-fish	FSOD, Fully Supervised
penguin-finder-seg	FSOD, Fully Supervised
pig-detection	FSOD, Fully Supervised
roboflow-trained-dataset	FSOD, Fully Supervised
sea-cucumbers-new-tiles	FSOD, Fully Supervised
thermal-cheetah	FSOD, Fully Supervised
tomatoes-2	FSOD, Fully Supervised
trail-camera	FSOD, Fully Supervised
underwater-objects	FSOD, Fully Supervised
varroa-mites-detection–test-set	FSOD, Fully Supervised
wb-prova	FSOD, Fully Supervised
weeds4	FSOD, Fully Supervised

Industrial	Link
-grccs	FSOD, Fully Supervised
13-lkc01	FSOD, Fully Supervised
2024-frc	FSOD, Fully Supervised
aircraft-turnaround-dataset	FSOD, Fully Supervised
asphaltdistressdetection	FSOD, Fully Supervised
cable-damage	FSOD, Fully Supervised
conveyor-t-shirts	FSOD, Fully Supervised
dataconvert	FSOD, Fully Supervised
deeppcb	FSOD, Fully Supervised
defect-detection	FSOD, Fully Supervised
fruitjes	FSOD, Fully Supervised
infraredimageofpowerequipment	FSOD, Fully Supervised
ism-band-packet-detection	FSOD, Fully Supervised
l10ul502	FSOD, Fully Supervised
needle-base-tip-min-max	FSOD, Fully Supervised
recode-waste	FSOD, Fully Supervised
screwdetectclassification	FSOD, Fully Supervised
smd-components	FSOD, Fully Supervised
truck-movement	FSOD, Fully Supervised
tube	FSOD, Fully Supervised
water-meter	FSOD, Fully Supervised
wheel-defect-detection	FSOD, Fully Supervised

Document	Link
activity-diagrams	FSOD, Fully Supervised
all-elements	FSOD, Fully Supervised
circuit-voltages	FSOD, Fully Supervised
invoice-processing	FSOD, Fully Supervised
label-printing-defect-version-2	FSOD, Fully Supervised
macro-segmentation	FSOD, Fully Supervised
paper-parts	FSOD, Fully Supervised
signatures	FSOD, Fully Supervised
speech-bubbles-detection	FSOD, Fully Supervised
wine-labels	FSOD, Fully Supervised

Medical	Link
canalstenosis	FSOD, Fully Supervised
crystal-clean-brain-tumors-mri-dataset	FSOD, Fully Supervised
dentalai	FSOD, Fully Supervised
inbreast	FSOD, Fully Supervised
liver-disease	FSOD, Fully Supervised
nih-xray	FSOD, Fully Supervised
spinefrxnormalvindr	FSOD, Fully Supervised
stomata-cells	FSOD, Fully Supervised
train	FSOD, Fully Supervised
ufba-425	FSOD, Fully Supervised
urine-analysis1	FSOD, Fully Supervised
x-ray-id	FSOD, Fully Supervised
xray	FSOD, Fully Supervised

Aerial	Link
aerial-airport	FSOD, Fully Supervised
aerial-cows	FSOD, Fully Supervised
aerial-sheep	FSOD, Fully Supervised
apoce-aerial-photographs-for-object-detection-of-construction-equipment	FSOD, Fully Supervised
electric-pylon-detection-in-rsi	FSOD, Fully Supervised
floating-waste	FSOD, Fully Supervised
human-detection-in-floods	FSOD, Fully Supervised
sssod	FSOD, Fully Supervised
uavdet-small	FSOD, Fully Supervised
wildfire-smoke	FSOD, Fully Supervised
zebrasatasturias	FSOD, Fully Supervised

Sports	Link
actions	FSOD, Fully Supervised
aerial-pool	FSOD, Fully Supervised
ball	FSOD, Fully Supervised
bibdetection	FSOD, Fully Supervised
football-player-detection	FSOD, Fully Supervised
lacrosse-object-detection	FSOD, Fully Supervised

Other	Link
buoy-onboarding	FSOD, Fully Supervised
car-logo-detection	FSOD, Fully Supervised
clashroyalechardetector	FSOD, Fully Supervised
cod-mw-warzone	FSOD, Fully Supervised
countingpills	FSOD, Fully Supervised
everdaynew	FSOD, Fully Supervised
flir-camera-objects	FSOD, Fully Supervised
halo-infinite-angel-videogame	FSOD, Fully Supervised
mahjong	FSOD, Fully Supervised
new-defects-in-wood	FSOD, Fully Supervised
orionproducts	FSOD, Fully Supervised
pill	FSOD, Fully Supervised
soda-bottles	FSOD, Fully Supervised
taco-trash-annotations-in-context	FSOD, Fully Supervised
the-dreidel-project	FSOD, Fully Supervised