Getting to the Point: Pointing Improves LVLMs at Counting

Anonymous ACL submission

Abstract

Pointing-based methods decompose complex tasks as sequential grounding and reasoning steps. Given a query, the model first grounds the relevant objects by generating their coordinates, and then predicts an answer conditioned on these points. While this approach has been shown to increase the performance of Large Vision-Language Models (LVLMs), it remains unclear why and how it improves the models' visual reasoning. In this work, we evaluate pointing-based methods in the task of zero-shot counting in visual scenes. We experiment with multiple fine-tuning and training-free approaches on state-of-the-art LVLMs, and compare them with *Point-then-Count* (PtC), where models first generate point coordinates for the target objects and then predict their count. Our results show that PtC achieves the highest accuracy among the evaluated approaches, with predicted points correctly grounded in the image in more than 94% of cases (based on F1-score). Mechanistic analyses show that gains arise from spatial information encoded in the predicted coordinates. Nevertheless, grounding performance varies across image regions, revealing spatial biases. Finally, the results indicate that PtC improves out-of-distribution generalization on both synthetic and real data, suggesting the potential of coordinates to help LVLMs improve their counting skills[1].

1 Introduction

Large Vision-Language Models (LVLMs) have achieved competitive performance on several multimodal tasks, such as visual question answering, image captioning, and optical character recognition (Deitke et al., 2025; Bai et al., 2025; Wang et al., 2025). However, prior work has shown that LVLMs struggle on complex cognitive tasks that require spatial (Rizzoli et al., 2025; Kamath et al., 2023), temporal (Rahmanzadehgervi et al., 2024), and counting (Fu et al., 2025; Vo et al., 2025) capabilities. Moreover, while some tasks require explicit visual evidence (e.g., grounded image captioning), in other settings LVLMs do not ground their answer in the image (Zhang et al., 2025), making it difficult for humans to interpret.

To improve the accuracy and explainability of LVLMs, pointing-based methods (Deitke et al., 2025) have recently proposed decomposing complex tasks into two explicit sequential steps. Given a natural-language query about the objects in an image, the model first grounds the objects mentioned in the query by predicting their 2D coordinates in the image. Then, it generates an answer to the query, conditioned on these grounded coordinates. Recent studies have shown that pointing can improve final-task accuracy in counting (Deitke et al., 2025), spatial reasoning (Yang and Huang, 2025), robotic affordance prediction (Yuan et al., 2025), and document understanding (Ni et al., 2025). However, several key questions remain unresolved: i) Does pointing encourage learning skills (e.g., counting) rather than narrow tasks (e.g., counting in a fixed range); ii) Are coordinates grounded in the visual input and therefore viable as explanations; and iii) What mechanism makes pointing improve performance? Addressing these questions is crucial for developing models capable of learning skills and producing outputs that are reliable and interpretable.

In this work, we study pointing in the context of zero-shot counting in visual scenes (Dai et al., 2024a; Alghisi et al., 2025; Zhang and Wang, 2024), a cognitive task that serves as a benchmark for multi-step reasoning. We decompose counting into a two-step process: 1) grounding all target instances in the image that satisfy a natural language query, and 2) aggregating this information to return the number of target objects. This makes counting an ideal testbed for studying point-

[1] We release all the material: link removed for double-blind review process

ing, as the task naturally separates target grounding from subsequent aggregation. We evaluate four state-of-the-art LVLMs under multiple fine-tuning and training-free approaches, including direct counting (Acharya et al., 2019; Rahmanzadehgervi et al., 2024) and reasoning (Jaech et al., 2024), and compare them with *Point-then-Count* (PtC) (Deitke et al., 2025), where LVLMs generate the target objects' coordinates, followed by their count. To understand which approach better supports skill learning, we evaluate LVLMs' Out-of-Distribution (OOD) generalization using images containing more targets than were observed during fine-tuning. For PtC models, we also study whether objects' coordinates can serve as suitable visual explanations by assessing how grounded these coordinates are in the image and their contributions towards the final prediction. Finally, to mitigate data contamination, reduce exploitable correlations during testing and perform fine-grained analyses, we evaluate LVLMs by generating synthetic datasets, balanced across object categories, labels, and spatial configurations. To ensure that our findings extend beyond synthetic settings, we also evaluate model performance on real-world data.

We frame our work around three research questions:

1. **Does Pointing Improve Task Performance, Encouraging Skill Learning in Counting?** We assess four state-of-the-art LVLMs under multiple counting approaches and evaluate their performance across multiple settings. We find that point supervision improves performance in unseen scenarios with more target objects than those observed during fine-tuning on both synthetic and real-world data. Moreover, deriving the final count from the number of coordinates achieves the highest accuracy.

2. **Can Coordinates Serve as Valid and Accurate Visual Explanations?** By comparing predicted coordinates with ground-truth annotations on synthetic data, we show that these points are grounded in more than 94% of cases. However, grounding performance for some models is not uniform across the image, hinting at potential pre-training biases or limitations in the attention mechanism.

3. **Why Pointing Improves Counting Performance?** When we replace each coordinate with an irrelevant token, no model achieves the same OOD generalization performance on synthetic data, underscoring the importance of spatial information for generalization. Further ablations reveal that PtC models mostly rely on coordinates to compute the total number of objects, disregarding the visual modality.

Our findings suggest that incorporating pointing as an intermediate task changes what LVLMs learn: models trained to generate grounded coordinates appear to acquire a more general counting procedure and expose intermediate evidence that can be verified. At the same time, the remaining failure modes indicate substantial room for improvement, motivating training paradigms that explicitly decompose perception and reasoning to better support skill learning and more interpretable outputs.

2 Literature Review

Decomposing complex cognitive tasks into intermediate sub-tasks has been shown to benefit humans and animals (Krueger and Dayan, 2009), as well as deep learning models (Wies et al., 2023; Wei et al., 2022; Nye et al., 2022; Zhou et al., 2023; Sprague et al., 2025). In the vision-language domain, decomposition has been adopted to separate *perception* from *reasoning* (Yi et al., 2018; Andreas et al., 2016). Pointing-based methods (Deitke et al., 2025) have been proposed as an intermediate subtask for LVLMs, and have proven effective in counting (Deitke et al., 2025; Clark et al., 2026), document understanding (Ni et al., 2025), spatial reasoning (Yang and Huang, 2025) and visual question answering (Man et al., 2025). However, prior work indicates that LVLMs produce intermediate groundings that are poorly aligned and may contain factual errors (Wu et al., 2025; Zhang et al., 2025), motivating evaluation protocols that focus on the intermediate evidence (Xia et al., 2025). Although datasets with intermediate grounding annotations exist (Lei et al., 2020; Wu et al., 2025), they may exhibit positional biases (Kirillov et al., 2023), thus limiting their usefulness for evaluation.

Counting requires a model to output the total number of target objects in an image. While early work focused on counting fixed target objects (Liang et al., 2022; Xie et al., 2018; Lin et al., 2022), recent approaches generalize across classes by specifying the target in a few-shot (Jeon et al., 2025; Ranjan et al., 2021; He et al., 2024; You et al., 2023) or zero-shot manner (Liu et al., 2025; Paiss et al., 2023; Jiang et al., 2023; Dai et al.,

2024b; Campbell et al., 2024). While LVLMs are often (pre-)trained on multiple counting benchmarks (Deitke et al., 2025; Beyer et al., 2024), recent work shows that they can struggle even with small counts (Alghisi et al., 2025; Rahmanzadehgervi et al., 2024), suggesting limited generalization. PtC alleviates this by providing coordinates as intermediate supervision (Deitke et al., 2025), yielding higher accuracy.

3 Experimental Setting

We investigate the role of pointing in the zero-shot counting task on four state-of-the-art LVLMs. Unlike prior studies that primarily emphasize accuracy (Deitke et al., 2025; Clark et al., 2026; Bai et al., 2025), our work focuses on explaining why coordinates help and what is the mechanism behind these gains. We evaluate each model on synthetic and real-world data across multiple counting approaches, comparing them with PtC in both in-distribution (ID) and out-of-distribution (OOD) settings to identify which method performs best and understand why.

3.1 Task: Zero-Shot Counting

Counting requires determining the number of instances of a target object in an image. In a zero-shot setting (Beyer et al., 2024; Deitke et al., 2025; Zhang and Wang, 2024; Campbell et al., 2024), the target object is specified via a natural-language query. Formally, given:

- an image $\mathcal{I} \in \mathbb{R}^{H \times W \times C}$ with height H, width W, and C channels depicting some objects;
- a query q in natural language that specifies the **class** and (optionally) the <u>attributes</u> of the target object (Alghisi et al., 2025) (e.g., *How many <u>blue</u> **stars** are there?*);
- a model $\mathcal{M}$, capable of processing an image $\mathcal{I}$ and a query q in natural language.

we assess the counting capabilities of $\mathcal{M}$ by comparing the generated answer $\hat{y} = \mathcal{M}(\mathcal{I}, q)$ with y, where $y \in \mathbb{N}_0$ corresponds to the number of targets objects in $\mathcal{I}$ that match the class and attributes specified in q. To ensure $\hat{y}$ follows our evaluation format, we employ a regex to remove any additional tokens (e.g., "The answer is…", or "Answer: …") and match the first valid answer. Further details are provided in Section A.4.

3.2 Synthetic Data

Existing datasets are often heavily imbalanced, both in terms of classes (Alghisi et al., 2025) (e.g., "person" frequently dominating the distribution) and counts (Al-Tahan et al., 2024) (small numbers occur more often than larger ones). Most exhibit strong positional bias, with targets frequently appearing in the center (Kirillov et al., 2023), thereby limiting our understanding of their grounding capabilities. Curated subsets (Deitke et al., 2025; Beyer et al., 2024) are often small and span only a narrow range, making them unsuitable to test OOD generalization. Finally, widely used real (Acharya et al., 2019) and synthetic (Johnson et al., 2017) datasets may have been included in the training data of some models, raising the risk of contamination.

We addressed these issues by generating tailored synthetic datasets with the CIVET framework (Rizzoli et al., 2025), which provides fine-grained control over image content, allowing us to specify the class, attributes, and position of each object. Leveraging such control, we construct multiple datasets, ensuring exhaustive coverage in terms of object types and position, and uniform label distribution. For each dataset, we define a sample as a tuple $(\mathcal{I}, q, \mathcal{C}, y)$, where:

- $\mathcal{I}$ is an image, which we subdivide into an $N \times M$ grid[2], where objects can be placed within each cell;
- q is a natural-language query specifying the class and attributes of the target object;
- $\mathcal{C} = \{(n, m) \mid 0 \leq n < N,\ 0 \leq m < M\}$ is the set of grid coordinates (n, m) of all target objects in $\mathcal{I}$;
- y is the number of target objects contained in $\mathcal{I}$ and corresponds to the cardinality of $\mathcal{C}$ (i.e., $y = |\mathcal{C}|$)

Having a grid-based representation gives us exact coordinates for each target object and control over the spatial layout. This makes the dataset ideal for training PtC models and for evaluating grounding without annotation errors.

We start by constructing a balanced dataset to count from one to nine, as prior work has shown that LVLMs exhibit limited counting ability even for small numerosities (Al-Tahan et al., 2024; Alghisi et al., 2025), and existing curated benchmarks

[2]For our experiments, we set $N = M = 9$.

adopt a similar scope (Beyer et al., 2024; Deitke et al., 2025). To reduce ambiguity, we consider images containing only target objects, as this ensures that model errors can be attributed to counting failures rather than confusion between targets and irrelevant objects. We report additional experiments with distractors in Section D.2.

For the training and validation splits, we use 10 target objects: six colored plusses and four white shapes. To obtain a uniform label distribution, we generate 81 unique samples (i.e., with a different spatial configuration) for each target object and count in the range $[1, 9]$. We then construct an `ID` test set using the same data-generation procedure as the training set, while varying both spatial configurations and target objects. Specifically, we consider 24 held-out target objects, obtained by recombining the 4 object classes and 6 colors observed during training into unseen color-shape combinations.

Finally, to understand which approach leads to better generalization in counting, we construct an `OOD` test set with images depicting from 10 to 18 objects. We generate this dataset using the same target objects as in the `ID` test set, allowing us to isolate generalization to higher counts from changes in object categories and visual attributes. Additional details on dataset construction and split sizes are provided in Section B.1.

3.3 Real-World Data

To assess whether our findings extend beyond synthetic data, we also train and evaluate our models on real-world images. As previously discussed, most existing counting benchmarks provide human annotations only over a limited count range (Deitke et al., 2025; Beyer et al., 2024), making them unsuitable for studying generalization to higher counts. We therefore adapt the OCID (Suchi et al., 2019) dataset for counting: OCID contains real-world scenes in which objects are placed incrementally, yielding images with 1 to 20 objects. Each object is associated with a ground-truth segmentation mask, from which we derive pointing coordinates $\mathcal{C}$ using the mask centroid. This makes OCID a well-suited real-world counterpart to our synthetic dataset, with a similar count range (1-20 instead of 1-18) and instance-level coordinates for evaluating pointing-based counting. Using images with 1 to 10 objects, we construct the train, validation, and `ID` test splits, and reserve images with 11 to 20 objects for an `OOD` test split. Further details on dataset construction and splits are provided in Section B.2.

3.4 Approach

We assess LVLM's capability to count under different approaches, and compare their performance on unseen scenarios to understand which approach generalizes best and why. Following the notation introduced in Sections 3.1 and 3.2, each model $\mathcal{M}$ takes as input an image $\mathcal{I}$ and a natural-language query q specifying the target object. We consider the following approaches:

- **Direct Counting (DC)** (Acharya et al., 2019): $\mathcal{M}$ directly predicts the number of target objects $\hat{y}$ based on $(\mathcal{I}, q)$.
- **Point-then-Count (PtC)** (Deitke et al., 2025): $\mathcal{M}$ first outputs the coordinates of the target objects and subsequently generates the predicted count $\hat{y}$.
- **Coordinates (# Coord.)**: taking inspiration from grounding/detection-based counting (Dai et al., 2024b; Huang et al., 2024), we consider a variant of PtC in which the final count is deterministically computed as the number of predicted coordinates (Sec. A.4).
- **List-then-Count (LtC)** (Hou et al., 2025): a variant of PtC in which $\mathcal{M}$ first enumerates the target objects, optionally grounding each instance with a short description or location cue, and then outputs the predicted count $\hat{y}$.
- **Reasoning** (Jaech et al., 2024): an extension of Chain-of-Thought (Wei et al., 2022) in which, without the need for examples, $\mathcal{M}$ generates an intermediate reasoning trace before producing the predicted count $\hat{y}$.

To better understand the contribution of pointing-based supervision in counting, we additionally fine-tune LVLMs to count under a DC and PtC approach. This allows us to compare models that are trained on the same set of image-question pairs, while differing only in whether they receive coordinate-based supervision. We ensure that performance differences can be attributed to the learning objective by using the same training protocol. For both approaches, models are optimized with the standard next-token prediction objective (Deitke et al., 2025) using LoRA (Hu et al., 2022). At inference time, we generate the answers with greedy decoding, using each LVLM's default stopping criterion. Additional details about training, inference, and hardware can be found in Section A.

Model	*Synthetic*		*Real-World*	
	DC	**PtC**	**DC**	**PtC**
Qwen2.5-VL 3B	98.11 (↑ 31.33)	**99.94** (↑ 33.15)	70.00 (↑ 27.62)	**81.90** (↑ 39.52)
Qwen2.5-VL 7B	93.02 (↑ 15.33)	**99.88** (↑ 22.19)	73.57 (↑ 19.76)	**76.19** (↑ 22.38)
LLaVA-OneVision	**99.81** (↑ 12.61)	99.76 (↑ 12.56)	69.05 (↑ 29.53)	**77.14** (↑ 37.62)
InternVL3.5	99.81 (↑ 13.20)	**100.00** (↑ 13.39)	74.52 (↑ 19.52)	**80.24** (↑ 25.24)

Table 1: Accuracy (%) on Synthetic and Real-World in-distribution (ID) data after DC or PtC fine-tuning. Values in parentheses denote absolute improvements over the corresponding pre-trained model. **Bold** values indicate the best result on synthetic and real-world, respectively. Overall, models benefit more from point supervision, suggesting PtC as a more robust approach across object counts. *Notation: **DC**: Direct Counting, **PtC**: Point-then-Count*

3.5 Models

To understand whether our findings generalize across LVLMs with different architectural and image pre-processing choices, we evaluate representative state-of-the-art models: Qwen2.5-VL 7B (Bai et al., 2025), LLaVA-OneVision-1.5 8B (An et al., 2025), and InternVL3.5 8B (Wang et al., 2025). In addition, we include Qwen2.5-VL 3B to study the effect of LLM scaling. We focus our analysis on open-source LVLMs, as closed-source models cannot be fine-tuned and may rely on external tools, making it difficult to assess their actual counting capabilities. LVLMs combine a vision encoder with an LLM (i.e., a text decoder) to perform vision-language tasks, but differ in how visual features are integrated into the language model. Both LLaVA-OneVision and InternVL3.5 employ a learned projection layer to map the visual information extracted by the encoder into the LLM embedding space, while in Qwen2.5-VL the visual representation is passed directly to the decoder. Regarding image pre-processing, InternVL3.5 applies dynamic tiling, splitting each image into non-overlapping 448×448 crops. In contrast, LLaVA-OneVision and Qwen2.5-VL support variable-resolution inputs, eliminating the need for explicit resizing or cropping. Additional details about the checkpoint used for each model can be found in Section A.1.

4 Experiments

We evaluate four state-of-the-art LVLMs to count under different approaches. To evaluate which approach encourages skill acquisition, we compare their performance in an OOD scenario. We further examine whether predicted coordinates can serve as valid and accurate visual explanations, and conduct a mechanistic analysis to understand how pointing influences counting behavior. Because all datasets are balanced across object types and count labels, we report our results using accuracy.

4.1 Does Pointing Always Improve Performance?

Although pointing has been shown to improve LVLMs' counting accuracy (Deitke et al., 2025), it is unknown whether point supervision always outperforms DC. Investigating this question is crucial for practical applications, as the additional computation time and resources required to generate target coordinates may outweigh modest accuracy gains. This analysis also provides useful information to select the appropriate fine-tuning strategy under varying accuracy-efficiency trade-offs. We thus study whether fine-tuning a model with point supervision leads to better performance in an ID setting, and compare its performance against DC.

The results in Table 1 show that models generally benefit more from point supervision than from DC fine-tuning. This trend is particularly noticeable in the real-world setting, where PtC improves accuracy by up to 11.9% over DC. On synthetic data, the advantage of PtC is less pronounced, with Qwen2.5-VL 3B, LLaVA-OneVision, and InternVL3.5 achieving comparable performance under the two fine-tuning strategies. However, additional per-count analysis in the appendix (Fig. 11) reveals that, after DC fine-tuning, the performance for both Qwen2.5-VL models degrades as the number of target objects increases, especially between 1 and 8 objects. This non-monotonic pattern is not observed for PtC, where performance remains more stable. In addition, models fine-tuned for PtC exhibit lower variability across object types and, in some cases, greater robustness to increasing numbers of distractors. We provide further analyses of compositional generalization and distractor robustness in Sections D.1 and D.2.

Overall, these findings indicate that point supervision improves in-distribution counting accuracy,

Model	*Training-Free*					*Fine-Tuning*		
	DC	**PtC**	**# Coord.**	**LtC**	**Reasoning**	**DC**	**PtC**	**# Coord.**
Qwen2.5-VL 3B	20.67	2.98	2.26	3.30	–	30.86	19.44	**80.20**
Qwen2.5-VL 7B	23.41	14.04	12.96	6.38	–	32.00	46.19	**94.96**
LLaVA-OneVision 8B	19.19	9.31	7.51	2.78	14.76	30.20	72.22	**92.18**
InternVL3.5 8B	46.76	24.38	16.04	13.28	20.32	45.52	97.33	**97.94**
Gemma 4 E4B	22.48	16.56	16.20	22.89	21.34	–	–	–

Table 2: Accuracy (%) on the synthetic out-of-distribution (OOD) setting across training-free and fine-tuning approaches. **Bold** values indicate the highest accuracy among fine-tuned approaches, while underlined indicates the highest among training-free ones. Overall, computing the count from the coordinates predicted by fine-tuned models yields the highest accuracy. *Notation:* ***DC****: Direct Counting,* ***PtC****: Point-then-Count,* ***LtC****: List-then-Count*

especially on real-world images. Moreover, even when DC and PtC achieve similar aggregate performance on synthetic data, PtC yields more stable behavior as the number of objects increases, making it a more reliable fine-tuning strategy for counting.

4.2 Which Approach Encourages Skill Learning?

Understanding how models behave in unseen scenarios is particularly important for deployment, where the number of target objects may exceed what the model observed during training. In addition, investigating which approach leads to higher generalization is crucial for training, as collecting balanced real-world data for high object counts is almost unfeasible. To this end, we assess which approach encourages skill learning by measuring their performance in an OOD setting, using images with at most twice the targets seen during training.

We first evaluate the approaches on synthetic data. Since Reasoning is time- and compute-intensive, evaluation on the full dataset is unfeasible. We therefore compare all approaches on a stratified subset of the data (11%), and report results on the full OOD set for the best-performing techniques in the appendix (Tab. 13). Results in Table 2 show that fine-tuning surpasses training-free approaches. In general, models fine-tuned under PtC achieve higher accuracy than those trained on DC, suggesting that pointing may teach a more robust counting strategy, capable of generalizing to higher counts. Interestingly, deriving the final count from the number of coordinates (# Coord.) increases accuracy substantially, with 7 and 8B models exceeding 90%. Qwen2.5-VL 3B also exhibits a large improvement, but its performance remains below larger models, suggesting that capacity may be important.

Regarding training-free approaches, prompting models to point (PtC and # Coord) or enumerate (LtC) objects generally underperform DC. This suggests that, without task-specific fine-tuning, most LVLMs struggle to use grounding or enumeration as intermediate steps for counting, consistent with previous findings in dialogue tasks (Zhang et al., 2025). A similar trend emerges for Reasoning, where InternVL3.5 achieves less than half of its DC accuracy, indicating that reasoning traces do not necessarily improve counting. To assess whether this limitation is specific to older models, we additionally evaluate Gemma 4, released in April 2026. Even for this newer model, Reasoning achieves performance comparable to DC, but requires substantially more generated tokens. Unlike the other models, Gemma slightly benefits from LtC, suggesting that newer LVLMs may be capable of exploiting intermediate outputs, even without fine-tuning.

We next examine whether these findings transfer to real-world data, focusing on fine-tuned models, which achieved the highest accuracy on the synthetic benchmark. Results in Table 3 show that, except for Qwen2.5-VL 3B, fine-tuning improves over the corresponding training-free DC baseline (shown in gray). Consistent with the synthetic setting (Tab. 2), deriving the final count from the coordinates yields the largest gains, with Qwen2.5-VL 3B improving by $+29.03$ accuracy points over its fine-tuned DC counterpart. The performance gap with the synthetic setting is likely due to the greater complexity of real-world images, where objects may be stacked or partially occluded, and to the lack of spatial balance in the data. Interestingly, despite using a similar image pre-processing to Qwen2.5-VL and achieving high accuracy in the synthetic setting, coordinates provide a marginal improvement for LLaVA-OneVision, suggesting that it may require different fine-tuning conditions.

These results suggest the potential of point supervision to generalization to higher object counts.

Model	*Fine-tuning*			
	DC	PtC	# Coord.	DC
Qwen2.5-VL 3B	2.15	0.98	**31.18**	2.35
Qwen2.5-VL 7B	8.82	25.68	**33.53**	6.47
LLaVA-OneVision	12.55	15.69	**16.27**	1.76
InternVL3.5	18.82	25.88	**30.98**	8.24

Table 3: Accuracy (%) on the real-world out-of-distribution (OOD) setting for fine-tuning approaches as well as training-free direct counting (DC). Similar to its synthetic counterpart, using the coordinates to count yields the highest accuracy.

Model	F1	EM	Cons.
Qwen2.5-VL 3B	94.52	76.07	20.56
Qwen2.5-VL 7B	98.59	88.05	48.79
LLaVA-OneVision	96.91	80.88	77.72
InternVL3.5	99.89	98.06	99.27

Table 4: Coordinate reliability (%) on the synthetic out-of-distribution (OOD) setting, reported using F1-score, exact match (EM), and consistency (Cons.) Consistency varies substantially across models, showing that, despite achieving high F1-scores, the number of predicted coordinates does not always match the model's final count.

Differing from prior work (Deitke et al., 2025), we observe that using the predicted coordinates to count can be advantageous beyond the training distribution, as the model's final answer is often inconsistent with the number of predicted points.

4.3 Are Coordinates Valid and Accurate Visual Explanations?

While our results show that coordinates can improve LVLM accuracy, especially in the OOD setting, their usefulness as visual explanations depends on whether they are grounded in the image and consistent with the final answer. We therefore evaluate their reliability using three metrics: F1-score, which captures both missed targets and hallucinated points; exact match (EM), which assesses whether the entire set of coordinates is correct; and consistency (Cons.), which measures whether the number of coordinates matches the final count.

We focus on synthetic data as it is balanced across spatial positions, enabling more fine-grained analyses. We report results on the synthetic OOD setting in Table 4, and discuss the remaining settings in Section D.3. Results show that consistency varies substantially across models, indicating that the number of predicted coordinates often does not match the model's final count. This explains why deriving the count directly from the predicted coordinates outperforms PtC in Table 2. EM further shows that high counting accuracy does not always imply fully correct grounding. For example, Qwen2.5-VL 7B reaches 94% accuracy in Table 2, but only 88% EM, meaning that in roughly 6% of cases at least one predicted coordinate is incorrect. Nevertheless, F1-score remains high across all models, suggesting that grounding errors typically affect only a small subset of target objects.

We then perform a fine-grained analysis by computing the F1-score for each cell in our 9×9 grid. Figure 1 shows that both Qwen2.5-VL models and LLaVA-OneVision exhibit spatial biases, with higher F1-score in the left portion of the image. Since LLaVA-OneVision uses Qwen3 as its language backbone, this pattern may indicate a bias shared across this model family. In contrast, InternVL3.5 maintains near-uniform and almost perfect F1-score across the image, which may be due to its pre-training or dynamic tiling strategy.

Overall, our findings show that predicted coordinates are grounded in the image in more than 94% of cases, as measured by F1-score, supporting their use as visual explanations. However, the presence of spatial biases and reduced exact-match performance indicates that errors can concentrate in specific image regions, limiting the reliability of point-based explanations for some models.

4.4 Why does Pointing Improve Performance?

After showing that pointing improves LVLM counting accuracy and analyzing the reliability of the predicted coordinates, an important question remains: why does it help? Addressing this question is essential for understanding whether other tasks might also benefit from point supervision. To this end, we conduct a series of ablations to study why pointing improves LVLMs' counting accuracy, focusing on the synthetic OOD setting. We hypothesize that coordinates provide explicit spatial cues that help the model disambiguate among object instances. To test this, we fine-tune each model for PtC, but replace each coordinate with the token "X", and report results in Table 5a. Our results show that removing the coordinates consistently reduces accuracy, indicating that spatial information helps LVLMs' generalize to higher object counts. The largest drop occurs for Qwen2.5-VL 7B, which, despite achieving 97% on the ID setting (Section D.4), tends to repeatedly generate "X" tokens instead of meaningful outputs.

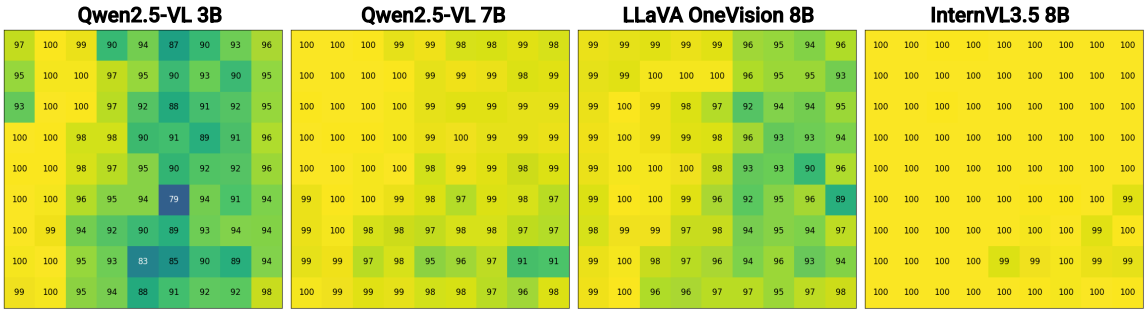

Figure 1: Cell-level F1-score (%) for each model across our 9×9 grid on the synthetic out-of-distribution (OOD) setting. Except for InternVL3.5, the other models exhibit a higher F1-score for the left part of the image, showing the presence of spatial biases.

Model	X-FT
Qwen2.5-VL 3B	↓ 60.50
Qwen2.5-VL 7B	↓ 91.42
LLaVA-OneVision 8B	↓ 75.78
InternVL3.5 8B	↓ 61.00

(a)

Ablation	
Image	Coord.
↓ 1.22	↓ 92.13
↑ 0.12	↓ 85.64
↓ 0.99	↓ 93.65
= 0.00	↓ 81.81

(b)

Table 5: Absolute change in accuracy under different ablations w.r.t. the PtC baseline. **(a)** reports the change on the synthetic out-of-distribution (OOD) setting (Table 2) after X-FT, where each coordinate is replaced with an "X" during PtC training. **(b)** reports the change on the synthetic in-distribution (ID) setting (Table 1) when ablating the input image or coordinates. Results highlight the importance of spatial information for generalization and the dominant role of textual coordinates.

To better understand how models leverage coordinates when counting, we conduct two additional ablations. First, we assess the model's reliance on coordinates by replacing $\mathcal{I}$ with a black image while providing ground-truth coordinates $\mathcal{C}$ as input. Second, we perform a leave-one-out ablation in which we remove one coordinate at a time and measure whether the model can recover the missing point. We also perform additional activation patching experiments to show each layer contribution, and report the results in Section D.5. We perform our ablations on the ID setting, where models achieve the highest accuracy (Tab. 1) and consistency (Tab. 10), making it well-suited to study the contribution of coordinates on the model's final count. Results in Table 5b reveal that removing the image has little effect ($< 2\%$ drop) on performance, indicating that models generate the final count mostly based on the coordinates, disregarding the visual modality. Ablating the coordinates leads to a significant drop in accuracy ($> 81\%$), indicating that models cannot recover when coordinate information is removed. This behavior resembles recent findings on reasoning models, where intermediate "thinking" tokens strongly influence the final prediction (Xu et al., 2026). This suggests that, in PtC, coordinates may play a similar role.

Our mechanistic analysis indicates that spatial information helps LVLMs to generalize to higher object counts. Moreover, we find that models rely primarily on the textual coordinates to produce the final count, validating the idea that reasoning over a structured representation may be easier than using raw visual features (Yi et al., 2018; Man et al., 2025).

5 Conclusion

We investigated the role of pointing-based methods in the zero-shot counting task by fine-tuning state-of-the-art LVLMs and comparing their performance with training-free methods. Our results show that Point-then-Count (PtC) achieves higher OOD generalization, suggesting that coordinate-based supervision can help LVLMs improve their counting skills. The generated coordinates are grounded in over 94% of cases, but fine-grained analyses reveal spatial biases. Ablations show that the spatial information in the coordinates helps generalize to higher object counts. Moreover, we observe that PtC models primarily rely on the textual coordinates to count, disregarding the image content. These findings motivate future training paradigms that more explicitly integrate grounding in reasoning, encouraging models to use visual evidence not only as an intermediate output but also as a reliable basis for subsequent reasoning. In the future, we plan to extend our results to other tasks and explore novel training strategies that better integrate grounding with visual reasoning.

6 Limitations

We focus our analysis on open-source LVLMs up to 8B parameters, since proprietary models cannot be fine-tuned, and larger models exceed our computational resources. Our study is also limited to the zero-shot counting task: understanding whether the findings hold in other vision-language tasks should be addressed in future work. In addition, although we validate our findings on real-world images, OCID serves as a relatively controlled scenario. Further evaluation is needed to assess whether these methods scale to more complex settings, including scenes with more variability, images containing a higher number of objects, and more diverse referring expressions. Finally, our OOD evaluation focuses on one specific aspect of counting skill: generalization to higher object counts than those observed during training. This allows us to assess whether coordinate-based supervision improves count-range extrapolation, while leaving broader forms of counting generalization to future work.

References

Manoj Acharya, Kushal Kafle, and Christopher Kanan. 2019. Tallyqa: Answering complex counting questions. In *Proceedings of the AAAI conference on artificial intelligence*, pages 8076–8084.

Haider Al-Tahan, Quentin Garrido, Randall Balestriero, Diane Bouchacourt, Caner Hazirbas, and Mark Ibrahim. 2024. Unibench: Visual reasoning requires rethinking vision-language beyond scaling. In *The 38th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, pages 82411–82437.

Simone Alghisi, Gabriel Roccabruna, Massimo Rizzoli, Seyed Mahed Mousavi, and Giuseppe Riccardi. 2025. [delre] constructing vlms' reasoning in counting. *arXiv preprint arXiv:2510.19555*.

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Didi Zhu, and 1 others. 2025. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, and 1 others. 2024. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*.

Declan Campbell, Sunayana Rane, Tyler Giallanza, Camillo Nicolò De Sabbata, Kia Ghods, Amogh Joshi, Alexander Ku, Steven Frankland, Tom Griffiths, Jonathan D Cohen, and 1 others. 2024. Understanding the limits of vision language models through the lens of the binding problem. *Advances in Neural Information Processing Systems*, 37:113436–113460.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinuo Yang, and 1 others. 2026. Molmo2: Open weights and data for vision-language models with video understanding and grounding. *arXiv preprint arXiv:2601.10611*.

Siyang Dai, Jun Liu, and Ngai-Man Cheung. 2024a. Referring expression counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16985–16995.

Siyang Dai, Jun Liu, and Ngai-Man Cheung. 2024b. Referring expression counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16985–16995.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, and 31 others. 2025. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 91–104.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, and 2 others. 2025. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24108–24118.

Hosein Hasani, Amirmohammad Izadi, Fatemeh Askari, Mobin Bagherian, Sadegh Mohammadian, Mohammad Izadi, and Mahdieh Soleymani Baghshah. 2025. Understanding counting mechanisms in large language and vision-language models. *arXiv preprint arXiv:2511.17699*.

Jinghui He, Bo Liu, Fan Cao, Jian Xu, and Yanshan Xiao. 2024. Few-shot object counting with dynamic

similarity-aware in latent space. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–14.

Kuinan Hou, Jing Mi, Marco Zorzi, Lamberto Ballan, and Alberto Testolin. 2025. Assessing the visual enumeration abilities of specialized counting architectures and vision-language models. *arXiv preprint arXiv:2512.15254*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhizhong Huang, Mingliang Dai, Yi Zhang, Junping Zhang, and Hongming Shan. 2024. Point segment and count: A generalized framework for object counting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17067–17076.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Yerim Jeon, Subeen Lee, Jihwan Kim, and Jae-Pil Heo. 2025. Mutually-aware feature learning for few-shot object counting. *Pattern Recognition*, 161:111276.

Ruixiang Jiang, Lingbo Liu, and Changwen Chen. 2023. Clip-count: Towards text-guided zero-shot object counting. In *Proceedings of the 31st ACM International Conference on Multimedia*, page 4535–4545.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Amita Kamath, Jack Hessel, and Kai-Wei Chang. 2023. What's "up" with vision-language models? investigating their struggle with spatial reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9161–9175.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, and 1 others. 2023. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026.

Kai A. Krueger and Peter Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394.

Jie Lei, Licheng Yu, Tamara Berg, and Mohit Bansal. 2020. TVQA+: Spatio-temporal grounding for video question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8211–8225, Online. Association for Computational Linguistics.

Dingkang Liang, Wei Xu, and Xiang Bai. 2022. An end-to-end transformer model for crowd localization. In *European Conference on Computer Vision*, pages 38–54.

Hui Lin, Zhiheng Ma, Rongrong Ji, Yaowei Wang, and Xiaopeng Hong. 2022. Boosting crowd counting via multifaceted attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19628–19637.

Shuai Liu, Peng Zhang, Shiwei Zhang, and Wei Ke. 2025. Countse: Soft exemplar open-set object counting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21536–21546.

Yunze Man, De-An Huang, Guilin Liu, Shiwei Sheng, Shilong Liu, Liang-Yan Gui, Jan Kautz, Yu-Xiong Wang, and Zhiding Yu. 2025. Argus: Vision-centric reasoning with grounded chain-of-thought. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14268–14280.

Minheng Ni, Zhengyuan Yang, Linjie Li, Chung-Ching Lin, Kevin Lin, Wangmeng Zuo, and Lijuan Wang. 2025. Point-rft: Improving multimodal reasoning with visually grounded reinforcement finetuning. *arXiv preprint arXiv:2505.19702*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2022. Show your work: Scratchpads for intermediate computation with language models.

Roni Paiss, Ariel Ephrat, Omer Tov, Shiran Zada, Inbar Mosseri, Michal Irani, and Tali Dekel. 2023. Teaching clip to count to ten. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3170–3180.

Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. 2024. Vision language models are blind. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 18–34.

Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. 2021. Learning to count everything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3394–3403.

Massimo Rizzoli, Simone Alghisi, Olha Khomyn, Gabriel Roccabruna, Seyed Mahed Mousavi, and Giuseppe Riccardi. 2025. CIVET: Systematic evaluation of understanding in VLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4462–4480, Suzhou, China. Association for Computational Linguistics.

Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann

Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations*.

Markus Suchi, Timothy Patten, David Fischinger, and Markus Vincze. 2019. Easylabel: A semi-automatic pixel-wise object annotation tool for creating robotic rgb-d datasets. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6678–6684.

An Vo, Khai-Nguyen Nguyen, Mohammad Reza Taesiri, Vy Tuong Dang, Anh Totti Nguyen, and Daeyoung Kim. 2025. Vision language models are biased: Counting legs of an animal is surprisingly hard. In *2nd AI for Math Workshop @ ICML 2025*.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, and 1 others. 2025. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Noam Wies, Yoav Levine, and Amnon Shashua. 2023. Sub-task decomposition enables learning in sequence to sequence tasks. In *The Eleventh International Conference on Learning Representations*.

Qiong Wu, Xiangcong Yang, Yiyi Zhou, Chenxin Fang, Baiyang Song, Xiaoshuai Sun, and Rongrong Ji. 2025. Grounded chain-of-thought for multimodal large language models. *arXiv preprint arXiv:2503.12799*.

Jiaer Xia, Bingkui Tong, Yuhang Zang, Rui Shao, and Kaiyang Zhou. 2025. Bootstrapping grounded chain-of-thought in multimodal llms for data-efficient model adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 208–217.

Weidi Xie, J Alison Noble, and Andrew Zisserman. 2018. Microscopy cell counting and detection with fully convolutional regression networks. *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, 6(3):283–292.

Zhongxing Xu, Chengzhi Liu, Qingyue Wei, Juncheng Wu, James Zou, Xin Eric Wang, Yuyin Zhou, and Sheng Liu. 2026. More thinking, less seeing? assessing amplified hallucination in multimodal reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Wenjie Yang and Zengfeng Huang. 2025. Poivre: Self-refining visual pointing with reinforcement learning. *arXiv preprint arXiv:2509.23746*.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Zhiyuan You, Kai Yang, Wenhan Luo, Xin Lu, Lei Cui, and Xinyi Le. 2023. Few-shot object counting with similarity-aware feature enhancement. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6315–6324.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. 2025. Robopoint: A vision-language model for spatial affordance prediction in robotics. In *Conference on Robot Learning*, pages 4005–4020. PMLR.

Chenhui Zhang and Sherrie Wang. 2024. Good at captioning bad at counting: Benchmarking gpt-4v on earth observation data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7839–7849.

Hao Zhang, Hongyang Li, Feng Li, Tianhe Ren, Xueyan Zou, Shilong Liu, Shijia Huang, Jianfeng Gao, Leizhang, Chunyuan Li, and Jainwei Yang. 2025. Llava-grounding: Grounded visual chat with large multimodal models. In *Computer Vision – ECCV 2024*, pages 19–35, Cham. Springer Nature Switzerland.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Experimental Details

We complement the main paper by providing additional implementation details.

A.1 Models

We evaluate the following open-source LVLMs available on HuggingFace:

- Qwen2.5-VL 3B
- Qwen2.5-VL 7B
- LLaVA-OneVision-1.5 8B
- InternVL3.5 8B

These models were selected because they represent state-of-the-art open-source LVLMs with different architectural choices and image preprocessing strategies, as discussed in Section 3.5 of the main paper. In addition, we conduct preliminary experiments with Molmo-7B-O and training-free experiments on Gemma 4 E4B.

A.2 Training

We fine-tune each LVLM to count under two approaches, DC and PtC. In **DC**, we optimize $\mathcal{M}$ to map the input pair $(\mathcal{I}, q)$ directly to the target count y. In **PtC**, we fine-tune $\mathcal{M}$ to first predict the coordinates $\mathcal{C}$ of the target objects based on $(\mathcal{I}, q)$, and subsequently generate the target count y. Specifically, we prepend $\mathcal{C}$ to the target count y, and optimize the model's parameters such that $\mathcal{M}(\mathcal{I}, q) =$ *"Coordinates:* $(n_1, m_1), (n_2, m_2), \ldots, (n_y, m_y)$. *Answer:* y*"*. We sort the coordinates in a fixed left-to-right, top-to-bottom order, which was shown to outperform a random ordering (Deitke et al., 2025).

We train each model using LoRA (Hu et al., 2022) (rank $r = 32$, scaling $\alpha = 64$), inserting adapters into the query, key, and value matrices of LLM, vision encoder, and modality projection layer (when present). Models are loaded in FP16 precision and optimized using AdamW with default parameters and a learning rate of 1×10^{-5} (which achieved the lowest validation perplexity in preliminary experiments). Training runs for up to 10 epochs, with early stopping (patience $= 2$) based on validation perplexity. All experiments are conducted on a single NVIDIA A100 (80 GiB) GPU with batch size 1, and images are processed using each model's default pre-processing pipeline. Under this configuration, each experiment requires at most two days.

A.3 Inference

During inference, answers are generated using greedy decoding with each LVLM's default stopping criterion. The only exception is Reasoning, for which we use each model's recommended generation hyperparameters, such as top-p, top-k, and temperature, as specified in the corresponding Hugging Face documentation.

Regarding the token budget, for fine-tuned models, we set `max_new_tokens` $= 5$ for DC and `max_new_tokens` $= 1{,}000$ for PtC, ensuring that each model can generate both coordinates and the final answer regardless of differences across tokenizers. For training-free approaches, we set `max_new_tokens` $= 3{,}000$ for PtC and LtC, and `max_new_tokens` $= 32{,}768$ for Reasoning, to accommodate the longer reasoning trace.

We report the prompts used for each training-free approach in Figures 2, 3, 4, 5 and 6. For Molmo, we use a prompt for PtC closely aligned with the one reported in its corresponding paper (Deitke et al., 2025), to ensure that the model outputs the objects' coordinates. For the other models, we evaluate three different prompts on a subset of the validation set and retain the one that achieves the best results

A.4 Answer and Coordinate Extraction

To extract the predicted count $\hat{y}$, we first convert textual numbers into digits (e.g., "one" $\rightarrow$ 1, "five" $\rightarrow$ 5, "twenty" $\rightarrow$ 20) and then apply a regular expression that extracts the first consecutive sequence of digits from the generated output. If no valid number can be matched, we return -1. For PtC models, the same procedure is applied after locating the *Answer:* field. Predicted grounding coordinates are extracted using a separate regular expression that matches tuples of the form (n, m) in the generated output, while discarding malformed pairs. The predicted coordinate set $\hat{\mathcal{C}}$ is evaluated against the ground-truth set $\mathcal{C}$ to assess grounding quality (Section 4.3). In the Coordinates approach (Section 3.4), we instead compute the predicted count directly as the number of extracted coordinates (i.e., $\hat{y} = |\hat{\mathcal{C}}|$).

B Dataset Construction

We complement the main paper by providing additional information about synthetic and real-world dataset construction.

Prompt for DC

```
Answer using as few words as possible.
```

Figure 2: Prompt used for training-free Direct Counting

Prompt for PtC (others)

```
Count the object(s) in the image. First generate the object's location(s) (returning a pair of
(x, y) coordinates between 0 and 100), then return the total number of objects. Only answer with
"Coordinates: ... Answer:".
```

Figure 3: Prompt used for training-free Point-then-Count for LLaVA-OneVision, InternVL3.5, and Qwen2.5-VL

Prompt for PtC (Molmo)

```
Count by pointing.
```

Figure 4: Prompt template used for training-free Point-then-Count for Molmo.

Prompt for LtC

```
You are given an image.
1. First, list all instances of the object mentioned in the question as a numbered list. Each
list item must contain a brief identifier (e.g., position or distinguishing detail).
2. Then, count the number of listed items.
3. Finally, provide the final answer within <answer></answer>. The answer must match the number
of listed items.

Output format:
<list>
1.
2.
</list>
<answer>N</answer>

Question:
```

Figure 5: Prompt used for List-then-Count

Prompt for Reasoning

```
Only answer with the numerical value.
```

Figure 6: Prompt used for Reasoning models.

B.1 Synthetic Datasets

As image size can affect model accuracy, we generate each image $\mathcal{I}$ at a resolution of 672×672 pixels, following prior findings (Rizzoli et al., 2025). For models that do not support this resolution, we apply their default image pre-processing strategy.

As discussed in Section 3.2, we subdivide each image into a 9×9 grid, where each cell corresponds to a 74×74 pixel region. Since $74 \times 9 = 666$, we add 3 pixels of padding on each side so that the grid is centered within the 672×672 image. Each object occupies an area of 64×64 pixels and can be placed in the center of a cell. To avoid occlusions, we place each object in a distinct cell. Within the grid, coordinates are represented as tuples (n, m), where n and m denote the row and column indices,

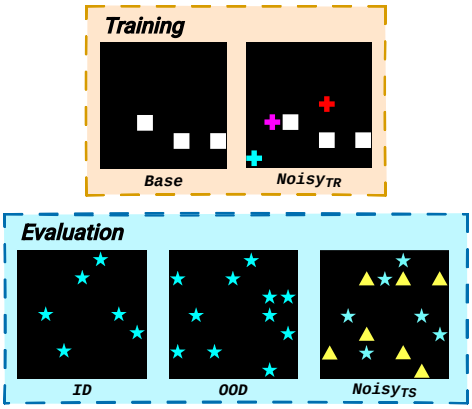

Figure 7: Visual overview of our synthetic datasets. In the main paper, we consider Base for training and ID and OOD for evaluation. Section D.2 discuss the results for Noisy$_{TR}$ and Noisy$_{TS}$

respectively. The top-left cell corresponds to $(0, 0)$, while the bottom-right cell corresponds to $(8, 8)$.

Following prior work (Alghisi et al., 2025), we mitigate the effect of external confounding factors by considering images with a uniform black background, reducing the chance that irrelevant elements are mistaken for targets, and by placing objects in distinct cells to avoid occlusions. To ensure that the proposed setting does not induce a severe distribution shift, we run a preliminary experiment to assess the performance of LVLMs on the generated images and discuss the results in Section C.

We generate images incrementally to minimize differences between images containing i and $i-1$ target objects. Specifically, each image $\mathcal{I}_i$ is obtained by adding one target object to $\mathcal{I}_{i-1}$ (leaving the other objects unchanged). This ensures that consecutive images differ by exactly one object, allowing us to study how increasing the number of objects affects LVLMs while minimizing variation due to object position. We use the same procedure when introducing distractors into the scene.

Queries q are generated using the template *"How many* <color> <shape>*s are there?"*, where the placeholders are replaced with the color and shape of the target object (e.g., "How many blue stars are there?").

We describe our training and evaluation datasets below and provide a visual overview in Fig. 7.

Training data. As described in Section 3.2, we fine-tune models on images containing 1 to 9 target objects. To avoid overfitting on a specific class-attribute combination, we consider 10 different target objects: six colored plusses (i.e., *red*, *green*, *blue*, *cyan*, *magenta*, *yellow*) and four white shapes (i.e., *circle*, *square*, *star*, *triangle*).

The Base training setting, described in Section 3.2, contains 7,290 samples, obtained from 9 count labels, 10 target objects, and 81 spatial configurations. We split this dataset into 4,860 training samples and 2,430 validation samples.

We also construct a noisier training setting, Noisy$_{TR}$, to assess whether exposure to irrelevant objects during fine-tuning improves robustness. Starting from each image $\mathcal{I}$ in the Base setting, we add $d \in \{1, 2, 3\}$ distractors to randomly selected empty cells. Each distractor is sampled uniformly from the set of non-target objects. To avoid introducing reasoning shortcuts, we enforce a uniform distribution over both the number and type of distractors. The resulting dataset has the same size as Base, allowing us to isolate the effect of increased scene complexity on LVLM learning without confounding it with additional training data.

Evaluation data. The ID and OOD test sets follow the construction described in Section 3.2. Both contain 17,496 samples and use the same 24 held-out target objects, obtained from all combinations of 4 shapes and 6 colors. The ID set contains images with 1 to 9 target objects, while the OOD set contains images with 10 to 18 target objects.

We further construct a distractor-based test set, Noisy$_{TS}$, to evaluate robustness to irrelevant objects. To isolate the effect of distractors, we start from the ID images and add distractors while keeping the target count and target positions fixed. Since exhaustively evaluating all target-distractor pairs is computationally infeasible, we fix the target object to *blue star*, which achieves the highest counting accuracy in the ID setting (Section D.1). We use the remaining 23 class-attribute combinations as distractors, allowing us to marginalize over distractor types. To study the effect of distractor quantity, we divide the dataset into nine segments, where segment $d \in \{1, \ldots, 9\}$ contains images with exactly d distractors. For each image, we instantiate three distractor spatial configurations by placing the distractors in different empty cells. Each segment contains 50,301 samples, for a total of 452,709 samples.

B.2 Real-World Dataset

To evaluate whether our findings extend beyond synthetic images, we construct a real-world count-

Model	DC	PtC
Qwen2.5-VL 3B	**66.79**	45.71
Qwen2.5-VL 7B	77.69	**78.42**
LLaVA-OneVision 8B	**87.20**	62.91
InternVL3.5 8B	**86.61**	77.42
Molmo 7B	53.06	**99.95**

Table 6: Accuracy (%) of each on the synthetic `ID` setting for training-free DC and PtC. Values in **bold** indicate the higher result between the two approaches. Overall, all models achieve more than 66% accuracy, suggesting that our images are compatible with their (pre-)training distribution. On PtC, only Molmo achieves near-perfect accuracy, while other models either degrade or show little change.

	Circles	*Squares*	*Triangles*	*Stars*
R	99.86 ± 0.11	99.83 ± 0.34	99.97 ± 0.07	99.93 ± 0.08
G	99.90 ± 0.07	99.83 ± 0.21	99.93 ± 0.14	**100 ± 0.00**
B	99.86 ± 0.11	99.59 ± 0.40	99.93 ± 0.08	99.93 ± 0.08
C	99.90 ± 0.07	99.86 ± 0.19	99.97 ± 0.07	99.97 ± 0.07
M	99.97 ± 0.07	99.66 ± 0.29	99.97 ± 0.07	99.97 ± 0.07
Y	99.90 ± 0.07	99.79 ± 0.26	**100 ± 0.00**	99.97 ± 0.07

Table 7: Accuracy (%) across object shape and color combinations on the synthetic `ID` setting for fine-tuned PtC Models. Values report mean accuracy with standard deviation across models. Results in **bold** indicate the color-shape combination(s) with the highest accuracy and lower standard deviation. *Notation:* ***R****: red,* ***G****: green,* ***B****: blue,* ***C****: cyan,* ***M****: magenta,* ***Y****: yellow.*

ing benchmark from OCID (Suchi et al., 2019). OCID provides RGB images, instance-level segmentation masks, and scenes (i.e., sequences of images in which objects are incrementally added), allowing us to obtain images with different object counts under similar scene conditions.

We split OCID at the scene level to avoid overlap between train, validation, and test images. The split is stratified across floor and table scenes, and we include both top and bottom camera views when available. Images containing 1 to 10 objects are used for the train, validation, and `ID` test splits, yielding 1160, 200, and 420 samples, respectively. Images containing 11 to 20 objects are reserved for the `OOD` split, resulting in 510 samples.

For each image $\mathcal{I}$, we derive ground-truth coordinates $\mathcal{C}$ from the instance-level segmentation masks. For each object, we first compute the mask centroid and then select the valid mask pixel closest to it as the representative point. This ensures that the coordinate lies on the target object itself, avoiding cases where the centroid falls outside the visible mask or on an occluding object. Following prior work (Deitke et al., 2025), we normalize coordinates to the $[0, 100]$ range and keep one decimal digit. Since segmentation masks do not provide object-class labels, we use *"How many objects are there?"* as the query q.

Due to the limited size of the training split, we apply data augmentation to the training images (i.e., color jittering, random crops, translations, rotations, and resizing) while updating the corresponding normalized coordinates. This process yields a total of 12,760 training samples.

C Preliminary Results

To ensure that the setting discussed in Sections B.1 and 3.2 does not induce a severe distribution shift, we run a preliminary experiment to assess the performance of LVLMs on the generated images. The results in Table 6 show that all pre-trained models achieve more than 66% accuracy (best results between DC and PtC) on the `ID` setting, suggesting that our images are compatible with their pre-training distribution. When asked to first point to the objects, Molmo's performance almost doubles, reaching near-perfect accuracy and indicating that the model benefits considerably from point supervision. In contrast, the performance of Qwen2.5-VL 3B, LLaVA-OneVision, and InternVL3.5 decreases, suggesting that these models may not have been pre-trained to ground objects or to use such information for counting. The only exception is Qwen2.5-VL 7B, which achieves only a negligible gain in performance ($< 2\%$). While this may indicate that the model was trained to use grounding as an intermediate subtask (as mentioned in its report (Bai et al., 2025)), the predicted coordinates do not follow the prompting schema and are, in most cases, out of bounds, making them not viable as visual explanations.

D Additional Results & Analyses

We complement the main paper by providing additional results and analyses.

D.1 Compositional Generalization

We evaluate compositional generalization across object attributes by training the model on all individual shapes and colors while holding out specific shape-color combinations. At test time, the model is evaluated on these unseen combinations. Ta-

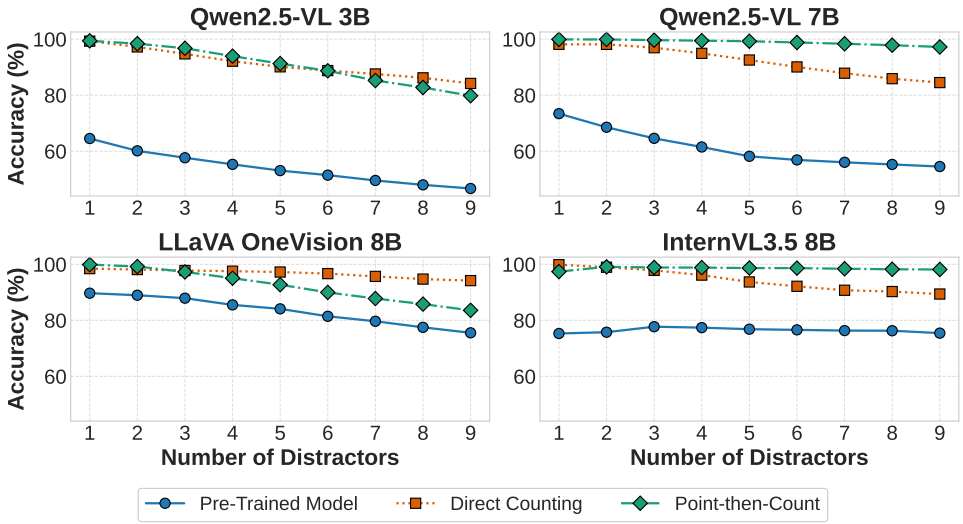

Figure 8: Accuracy (%) as a function of the number of distractors after fine-tuning on $\texttt{Noisy}_{\texttt{TR}}$ for DC or PtC. PtC maintains near-perfect accuracy for InternVL3.5 and Qwen2.5-VL 7B, while for Qwen2.5-VL 3B and LLaVA-OneVision PtC degrades faster than DC as distractors increase.

	Circles	*Squares*	*Triangles*	*Stars*
R	98.87 ± 1.06	96.85 ± 4.10	96.06 ± 7.61	99.31 ± 0.60
G	99.01 ± 1.18	95.58 ± 5.68	96.71 ± 5.88	98.94 ± 1.61
B	**99.69 ± 0.62**	96.26 ± 4.20	97.46 ± 4.04	99.66 ± 0.24
C	99.38 ± 0.69	95.71 ± 6.71	96.40 ± 6.66	99.25 ± 1.25
M	99.14 ± 0.76	93.90 ± 8.12	95.03 ± 7.47	99.35 ± 0.49
Y	99.01 ± 0.70	96.50 ± 4.80	97.67 ± 4.39	98.80 ± 1.86

Table 8: Accuracy (%) across object shape and color combinations on the synthetic ID setting for fine-tuned DC models. Values report mean accuracy with standard deviation across models. Results in **bold** indicate the color-shape combination(s) with the highest accuracy and lower standard deviation. *Notation:* ***R****: red,* ***G****: green,* ***B****: blue,* ***C****: cyan,* ***M****: magenta,* ***Y****: yellow.*

	Circles	*Squares*	*Triangles*	*Stars*
R	99.37 ± 0.88	98.34 ± 3.13	98.01 ± 5.41	99.62 ± 0.52
G	99.45 ± 0.91	97.70 ± 4.36	98.32 ± 4.22	99.47 ± 1.19
B	99.78 ± 0.42	97.93 ± 3.29	98.70 ± 2.95	**99.79 ± 0.22**
C	99.64 ± 0.53	97.79 ± 4.92	98.18 ± 4.76	99.61 ± 0.91
M	99.55 ± 0.67	96.78 ± 6.14	97.50 ± 5.55	99.66 ± 0.46
Y	99.45 ± 0.66	98.15 ± 3.61	98.83 ± 3.13	99.38 ± 1.37

Table 9: Accuracy (%) across object shape and color combinations on the synthetic ID setting averaged across fine-tuned DC and fine-tuned PtC models. Values report mean accuracy with standard deviation across models. Results in **bold** indicate the color-shape combination(s) with the highest accuracy and lower standard deviation. *Notation:* ***R****: red,* ***G****: green,* ***B****: blue,* ***C****: cyan,* ***M****: magenta,* ***Y****: yellow.*

bles 7 and 8 report the average accuracy with the standard deviation across models in the ID setting for DC and PtC, respectively. Overall, PtC yields stronger compositional generalization, with two unseen shape-color combinations reaching 100% accuracy. PtC also exhibits lower variability across object types, with *blue squares* being the most challenging case while still achieving 99.59% accuracy. By contrast, models fine-tuned with DC reach at most 99.69% accuracy, down to 93.90% (for *magenta squares*), and are more sensitive to variations in shape and color. For instance, performance on *Squares* differs by roughly 3 percentage points between *red* and *magenta*. DC also exhibits greater variability across models, with *magenta squares* showing the largest standard deviation at 8.12 percentage points. In contrast, PtC exhibits at most a variability of 0.4 percentage points, suggesting better transferability across models. Finally, we show in Table 9 that *blue stars* is the color-shape combination with the highest accuracy and lowest standard deviation across models and training approaches. This motivates our choice of *blue stars* as the target object for the $\texttt{Noisy}_{\texttt{TS}}$ setting (Section B.1).

Overall, our findings suggest that PtC promotes stronger compositional generalization across object types than DC.

D.2 Does Point Supervision Improve Robustness to Distractors?

To understand which approach is more robust to distractors, we fine-tune the models on a modified set that includes images with up to three distractors (i.e., $\texttt{Noisy}_{\texttt{TR}}$). We then evaluate their robust-

Model	ID			OOD			Noisy$_{TS}$		
	F1	**EM**	**Cons.**	**F1**	**EM**	**Cons.**	**F1**	**EM**	**Cons.**
Qwen2.5-VL 3B	99.91	99.59	100.00	94.52	76.07	20.56	93.59	69.76	99.56
Qwen2.5-VL 7B	99.89	99.55	100.00	98.59	88.05	48.79	99.19	95.77	99.75
LLaVA-OneVision 8B	99.52	98.31	99.98	96.91	80.88	77.72	89.59	70.57	99.82
InternVL3.5 8B	100.00	100.00	100.00	99.89	98.06	99.27	99.84	99.05	98.51

Table 10: Coordinate reliability (%) across three synthetic settings, reported using F1-score, exact match (EM), and consistency (Cons.). Results for Noisy$_{TS}$ are computed only on images containing 9 distractors.

Figure 9: Cell-level F1-score (%) for each model across our 9×9 grid on the synthetic ID setting.

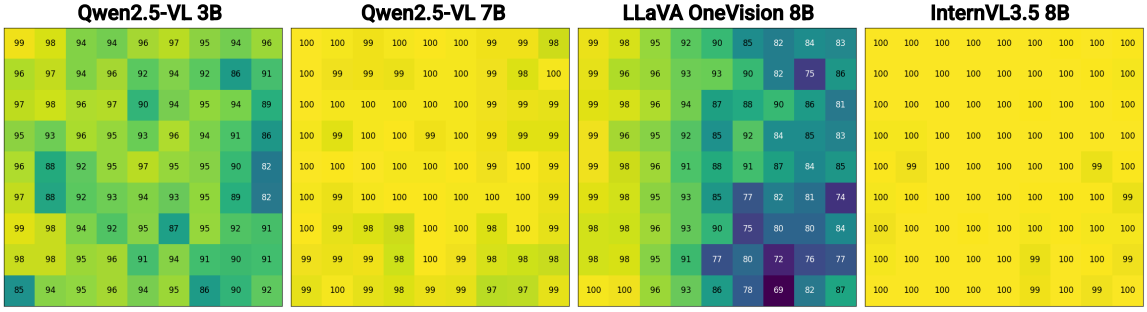

Figure 10: Cell-level F1-score (%) for each model across our 9×9 grid on the Noisy$_{TS}$ setting.

ness on images containing up to nine distractors (i.e., Noisy$_{TS}$) and report the results in Figure 8. Our plots show that pre-trained models become less accurate as the number of distractors increases, suggesting that they miscount in the presence of distractors. The only exception is InternVL3.5, which seems to be equally affected by 1 or 9 distactors. Both DC and PtC improve over the pre-trained models, but point supervision is particularly effective for InternVL3.5 and Qwen2.5-VL 7B, which maintain more stable results and achieve near-perfect accuracy even with nine distractors. For the remaining models, point supervision helps up to a moderate number of distractors, but degrades more quickly than DC once distractors become more numerous.

Overall, these results suggest that point supervision can yield more stable performance. However, generalization to a higher number of distractors depends on model architecture and/or (pre-)training data.

D.3 Grounding

We complement the main paper by reporting cell-level F1-scores for the remaining settings. We provide the F1-score, EM, and Consistency for all synthetic settings in Table 10. Results show that coordinates are especially reliable in the ID setting, since it matches the training distribution. Notably, most models achieve high and spatially consistent F1-scores across the entire image, as illustrated in Figure 9. The only exception is LLaVA-OneVision, which shows slightly reduced grounding performance near the image boundaries.

On Noisy$_{TS}$[3] results show that, despite each model being highly consistent, EM drops substantially for Qwen2.5-VL 3B and LLaVA-OneVision,

[3]We report results only for images with nine distractors, as this corresponds to the most challenging segment (see Fig. 8).

Model	ID	OOD	$\texttt{Noisy}_{\texttt{TS}}$
Qwen2.5-VL 3B	0.00	0.00	0.00
Qwen2.5-VL 7B	0.00	0.09	0.03
LLaVA-OneVision 8B	0.01	0.08	0.20
InternVL3.5 8B	0.00	0.00	0.00

Table 11: Percentage (%) of out-of-bounds coordinates predicted by each model across the synthetic ID, OOD, and $\texttt{Noisy}_{\texttt{TS}}$ settings. Results for $\texttt{Noisy}_{\texttt{TS}}$ are computed only on images containing 9 distractors.

Model	X-FT
Qwen2.5-VL 3B	98.11 (↓ 1.83)
Qwen2.5-VL 7B	97.37 (↓ 2.51)
LLaVA-OneVision 8B	99.71 (↓ 0.05)
InternVL3.5 8B	99.87 (↓ 0.13)

Table 12: Accuracy (%) of each model on the synthetic ID setting after X-FT, where each coordinate is replaced with an "X" during PtC training. Values in parentheses indicate the absolute change with respect to the corresponding PtC baseline (Table 1).

indicating that most failures stem from localization errors. Cell-level accuracy in Figure 10 reveals that LLaVA-OneVision exhibits the most pronounced spatial biases in this setting, with performance progressively decreasing from left to right (similar to Fig. 1). Qwen2.5-VL 3B shows a different trend and achieves a lower F1-score at the bottom and right edges. Qwen2.5-VL 7B exhibits a similar, yet less concerning, pattern, suggesting the possibility of a shared bias within this model family.

Since some models may occasionally produce coordinates that fall outside the image boundaries, which would lower the F1-score for specific spatial regions, we report the percentage of out-of-bounds predictions for each setting in Table 11. Overall, such errors are extremely rare, occurring in fewer than 0.21% of the cases across all models and settings. In particular, Qwen2.5-VL 3B and InternVL3.5 never produce out-of-bounds coordinates in any setting, while the remaining models only exhibit negligible rates. Interestingly, these errors appear almost exclusively in the more challenging OOD and $\texttt{Noisy}_{\texttt{TS}}$ settings, suggesting that distribution shifts and the presence of distractors slightly increase the likelihood of invalid coordinate predictions.

Overall, these results indicate that invalid coordinate predictions are not a significant source of error in our evaluation, validating the grounding trends reported in Figures 1, 9 and 10 and confirming that the variations in F1-score primarily reflect the presence of spatial biases.

D.4 Performance of X-FT on the ID setting

To show that models fine-tuned to output "X" tokens instead of coordinates can still count accurately on in-distribution examples, yet fail to generalize to object counts beyond those observed during fine-tuning (as shown in Table 5a), we evaluate them on the ID setting. The results in Table 12 show that, although models fine-tuned to output "X" instead of coordinates exhibit slightly lower performance than their PtC counterparts ($< 3\%$), all models still achieve more than 97% accuracy. This suggests that, in the ID setting, models can rely on coarse visual cues or on the textual pattern learned during fine-tuning to solve the task even without explicit spatial grounding. However, their accuracy drops substantially on the OOD setting, as shown in Table 5a. Taken together, these results indicate that replacing spatial information with a generic token preserves in-distribution performance, but degrades performance on higher object counts.

D.5 Layer-Wise Activation Patching

To further analyze when models rely on visual representations during counting, we perform a layer-wise activation patching experiment. The goal of this analysis is to identify at what layer substituting the visual representation of an input image (i.e., *source*) with that of a related image (i.e., *target*) stops affecting the model's output. Intuitively, if replacing the image representation at a given layer changes the answer towards the target image count, then the model is still using the information from the image. Conversely, if the substitution has no effect, the model is using the information aggregated in the textual tokens.

We construct a set of source-target pairs by stratified sampling from the synthetic ID dataset. Each source image contains N objects. For each source, we select target images depicting the same object category (e.g., "blue stars") but containing a different number of objects. The final set contains a total of 5,184 source-target pairs, computed as 9 count labels, 24 object types, 3 spatial configurations (i.e., source), and 8 related images (i.e., target).

For each source-target pair, we first run the model on the target input and cache the hidden states corresponding to the vision tokens at every transformer layer. We then run the model on the

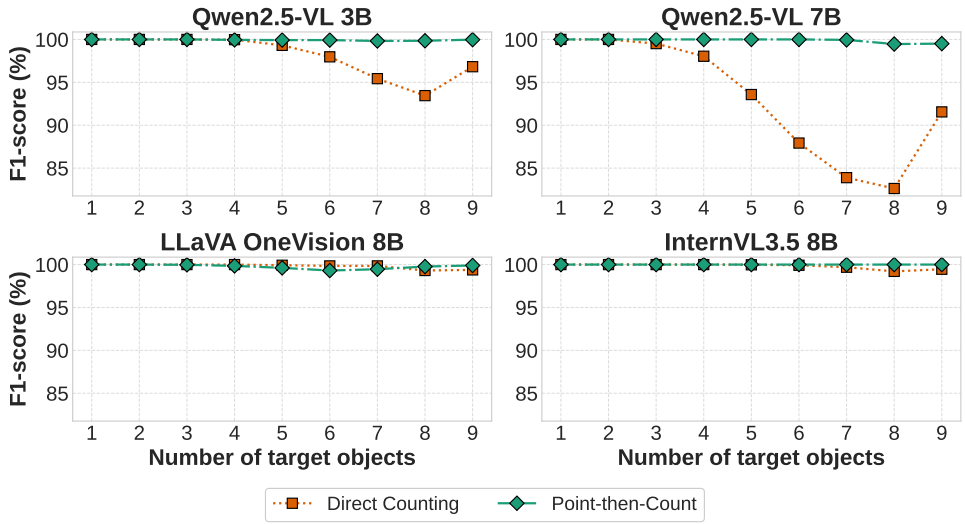

Figure 11: F1-score as a function of the number of target objects under the synthetic ID setting for models fine-tuned with DC or PtC. While LLaVA-OneVision and InternVL3.5 show similar performance, Qwen2.5-VL models benefit more from point supervision, suggesting PtC as a more robust choice across object counts.

source input. At each layer L, we path the source vision-token hidden states by replacing them with the cached target vision-token hidden states from the same layer, while leaving all other hidden states unchanged. The model then generates an answer from the patched source representation: this allows us to test whether injecting the target visual representation at a given layer changes the model's prediction.

We use this experiment to compare models fine-tuned on DC against models fine-tuned on PtC. Following prior work (Hasani et al., 2025), we report the results for each object count in the range one to nine. Figures 12, 13, 14 and 15 show, for each layer L, the percentage of outputs that match the original source answer. Low values (i.e., blue) indicate that replacing the source vision-token activations with the target changes the model prediction, suggesting that visual information is still influential at layer L. Conversely, high values (i.e., red) indicate that the model is unaffected by the patching operation, as it continues to produce the source answer.

Across all models and training approaches, we observe a consistent layer-wise transition. In early and middle layers, the percentage of source-matching outputs is close to zero, indicating that patching the target visual representation affects the model's answer. This suggests that the model is still relying on the image information. In contrast, in the final layers, the percentage of source-matching outputs rises sharply toward 100%, indicating that replacing the vision-token activations no longer changes the prediction. The exact layer for this transition varies across models and training approaches.

Model	*TF*	*Fine-Tuning*		
	DC	**DC**	**PtC**	**# Coord.**
Qwen2.5-VL 3B	21.64	31.66	19.52	**81.65**
Qwen2.5-VL 7B	22.94	31.88	46.54	**94.78**
LLaVA-OneVision	19.22	31.06	72.38	**92.11**
InternVL3.5	46.35	45.38	97.33	**98.05**
Gemma 4 E4B	23.58	–	–	–

Table 13: Accuracy (%) on the synthetic OOD setting across training-free (TF) and fine-tuning approaches. **Bold** values indicate the best result among fine-tuned approaches. Similar to the results in Table 2, computing the count from the coordinates predicted by fine-tuned PtC models yields the highest accuracy.

For Qwen2.5-VL 7B (Fig. 12), the transition occurs around layers 18 to 22. Under DC, the transition is more gradual and count-dependent, suggesting that counting an increasing number of object may require a larger number of layers, eventually exceeding the model's computational capacity (Hasani et al., 2025). In contrast, this transition happens approximately at the same layer for PtC, regardless of the object count. A similar pattern is visible for LLaVA-OneVision (Fig. 13) and InternVL3.5 (Fig. 14), but at different layers. The only exception seems to be Qwen2.5-VL 3B, which requires fewer layers when counting one object, but seems to use the same number of layers for higher counts, even under DC.

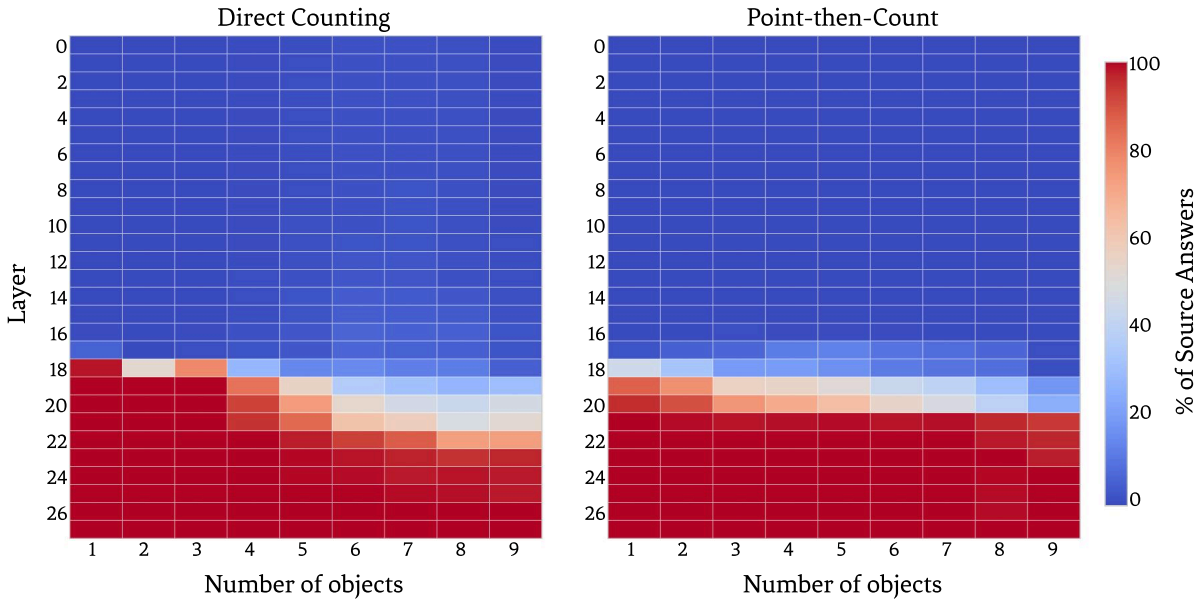

Figure 12: Layer-wise activation patching for Qwen2.5-VL 7B when fine-tuned for DC and PtC. Each cell shows the percentage of source answers after activation patching at layer L (y-axis) for images containing N objects (x-axis). We perform activation patching by substituting the hidden representations of the source image with those of a target image. Low values (blue) indicate that the patched visual representation changes the model's prediction, while high values (red) indicate that the model is unaffected by the patching and outputs the source answer.

Overall, activation patching reveals a common two-stage behavior across models. In the first stage, visual-token activations remain causally important. In the second stage, the answer becomes largely determined by information already integrated into the textual tokens. Regarding the training approach, PtC generally produces a sharper and more count-invariant transition, suggesting that it encourages the model to aggregate visual counting information into a potentially more stable intermediate representation before generating the final count.

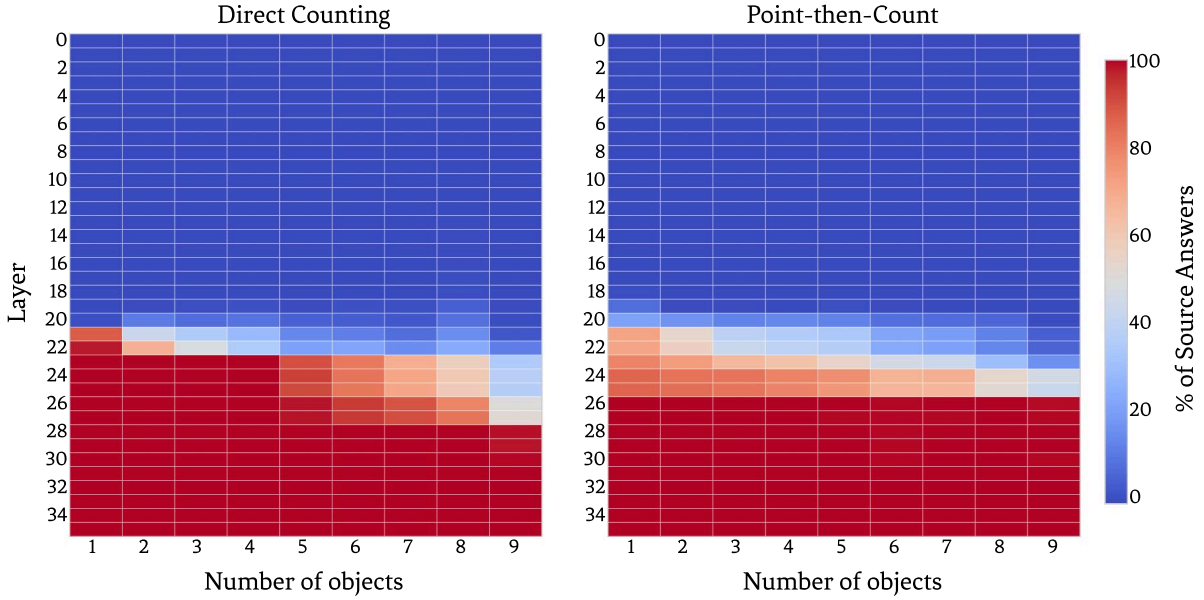

Figure 13: Layer-wise activation patching for LLaVA-OneVision when fine-tuned for DC and PtC. Each cell shows the percentage of source answers after activation patching at layer L (y-axis) for images containing N objects (x-axis). We perform activation patching by substituting the hidden representations of the source image with those of a target image. Low values (blue) indicate that the patched visual representation changes the model's prediction, while high values (red) indicate that the model is unaffected by the patching and outputs the source answer.

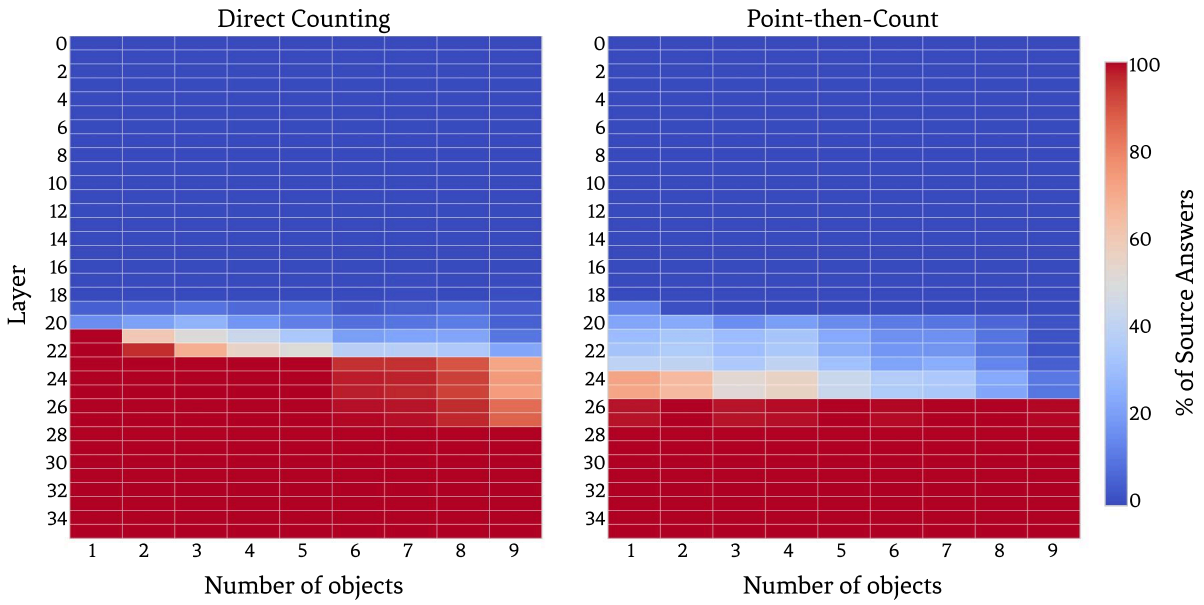

Figure 14: Layer-wise activation patching for InternVL3.5 when fine-tuned for DC and PtC. Each cell shows the percentage of source answers after activation patching at layer L (y-axis) for images containing N objects (x-axis). We perform activation patching by substituting the hidden representations of the source image with those of a target image. Low values (blue) indicate that the patched visual representation changes the model's prediction, while high values (red) indicate that the model is unaffected by the patching and outputs the source answer.

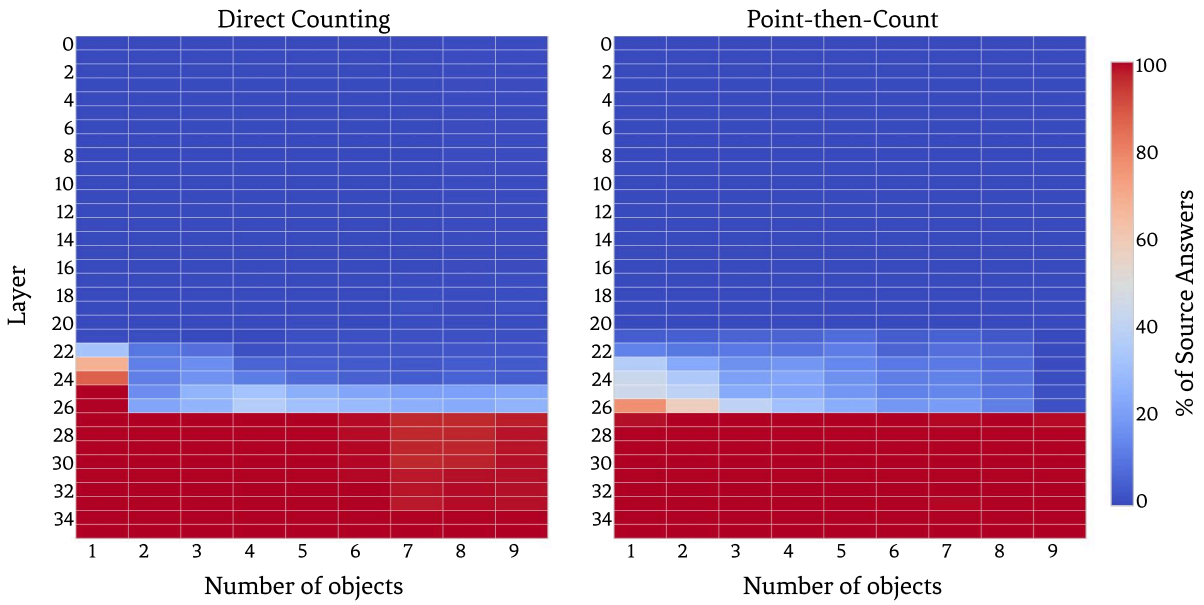

Figure 15: Layer-wise activation patching for Qwen2.5-VL 3B when fine-tuned for DC and PtC. Each cell shows the percentage of source answers after activation patching at layer L (y-axis) for images containing N objects (x-axis). We perform activation patching by substituting the hidden representations of the source image with those of a target image. Low values (blue) indicate that the patched visual representation changes the model's prediction, while high values (red) indicate that the model is unaffected by the patching and outputs the source answer.