Appendix

A. Implementation Details

A.1. Metric Details

We use symmetric mean percent error (sMAPE) as the primary metric for our benchmarks due to its resistance to bias for under/over predictions and small/large ground truths [25]. The standard metric for a counting benchmark is mean average error (MAE). MAE is popular, but heavily penalizes predictions that deviate by a small margin from big ground truths, highlighting the necessity for a metric that gives equal weighting to all questions. Mean average percent error (MAPE) initially seems appealing but is disproportionally inflated for small ground truths and is biased towards overpredictions. Mean square error (MSE) and root mean square error (RMSE) are also commonly used but are very sensitive to outliers because they square the error. Intuitively, performing well on almost all questions and poorly on a small subset should score better than consistently being wrong. Among commonly-used metrics, sMAPE is the only metric that evaluates performance in relation to the distribution of ground truth elements [11]. There are two common definitions [15] for sMAPE, but we use the one that scales to 100%. sMAPE is given by:

$$\text{sMAPE} = 100 \cdot \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|} \tag{2}$$

where y_i represents the actual values, $\hat{y}_i$ represents the predicted values, and n is the number of observations. sMAPE is capped at 100%, providing a finite scoring range. This feature is ideal for challenging tasks like ours, as it penalizes model responses that fail to produce an answer.

A.2. Output Tokens

To maximize the VLM's chance at success, we allocate a high number of output tokens to generate a rationale and output. This varies per model. We give 4000 tokens to InternVL2, 2000 tokens to Molmo, and 8192 tokens to Qwen2VL, following their max output lengths. For GPT-4o, we use the default of 4096 tokens.

B. CAPTURE Dataset Creation Details

The following expands upon Sec. 2.2. While FSC-147, a diverse counting dataset with manual annotations, is a strong starting point, it cannot immediately be adapted to our task. To make the task of amodal counting solvable, our dataset requires images with patterns in them. A person (or model) can infer how the pattern would continue and thus accurately predict the total number. For questions to be answerable, the dataset's images must be filtered down to represent patterns a model or person could recognize.

Our filtering process follows two stages. **First**, we prompt GPT-4o to determine whether the objects were arranged in a pattern. **Second**, if the model responded with "no", the images were immediately discarded. If the model output was "yes", the log probability of the token is stored. Empirically, we found that higher log probability values (i.e. higher confidence scores) corresponded to more well-defined patterns in the image. Thus, we use the log probabilities for filtering.

Specifically, let P_{yes} be the log probability of the "yes" token and T denote the threshold for determining how well-defined a pattern is.[2] To filter the images based on pattern rigidity, we apply the following condition: $e^{P_{\text{yes}}} \geq T$. This inequality yields 991 images from the original dataset (16.12%). Next, we manually filter each of the selected images to ensure that they indeed contain patterns and feature a countable number of objects, excluding 34 images. Afterward, we manually place a "fair" occluding box in each image, i.e. a box that leaves sufficient portions of the pattern visible, such that the pattern can still be inferred from the unoccluded portions of the image. Occluding boxes were also chosen with varying positions and sizes in the image.

C. Additional Analysis

Here we provide additional experiments that attempt to either increase model performance on CAPTURE or dissect the reasons behind poor model performance. Chain-of-Thought inhibits model performance, while temperature backoff slightly improves performance. Additionally, we find that models struggle at counting just occluded objects, are overconfident in occluded settings, and are biased to predict specific numbers.

C.1. Chain-of-Thought reduces model performance

Method	CAPTURE$^{\text{real}}$	CAPTURE$^{\text{synthetic}}$
GPT-4o	14.75	9.71
GPT-4o w/ CoT	14.94	7.73
Qwen2	29.33	11.74
Qwen2 w/ CoT	31.57	37.81

Table 6. CoT experiments (metric: sMAPE).

During development, we experimented with several common strategies including CoT. In Tab. 6, we find that CoT reduces model performance except in the occluded synthetic scenario, most likely because the included examples are very similar to the test prompt.

Model	Error (%) (↓)							
	Real				Synthetic			
	Unoccluded		Occluded		Unoccluded		Occluded	
	Original	w/ backoff (Δ)	Original	w/ backoff (Δ)	Original	w/ backoff (Δ)	Original	w/ backoff (Δ)
GPT-4o	13.34	12.57 (−0.77)	14.75	14.39 (−0.36)	5.90	5.93 (+0.03)	9.71	9.23 (−0.48)
InternVL2	26.17	27.09 (+0.92)	32.90	32.37 (−0.53)	16.44	15.59 (−0.85)	17.57	16.24 (−1.33)
Molmo	25.90	21.23 (−4.67)	32.49	28.17 (−4.32)	8.40	2.88 (−5.52)	17.73	15.85 (−1.88)
Qwen2VL	18.96	19.40 (+0.44)	29.33	28.47 (−0.86)	6.63	6.66 (+0.03)	11.74	11.51 (−0.23)
Avg. of 4 VLMs	21.09	20.07 (−1.02)	27.37	25.85 (−1.52)	9.34	7.76 (−1.58)	14.19	13.21 (−0.98)

Table 7. Comparison of models on CAPTURE across four scenarios (CAPTURE^{real} vs. CAPTURE^{synthetic}, Unoccluded vs. Occluded). "Original" indicates no backoff; "w/ backoff" indicates applying backoff, with Δ = (*w/ backoff*) − (*Original*). Negative Δ values indicate an improvement.

C.2. Temperature backoff slightly improves model performance

To improve VLM performance on CAPTURE, we address a trend we established during early testing. Most of the time, the VLM fails by reaching an incorrect answer. Sometimes, however, our benchmark can cause VLMs to produce a long and irrelevant response that strays from the original prompt, leading to the worst possible sMAPE score (100%).

To reduce the number of skipped questions, we experiment with *temperature backoff*, which iteratively decreases the sampling temperature. Because the answer extractor can immediately identify an incoherent output, we can regenerate the response with a lower temperature to get the model to answer the task properly. Consistent with our findings, Peeperkorn et al. [32] also finds that lower temperatures increase coherence in VLMs, thereby enhancing their chances of maintaining relevance to the prompt. Therefore, temperature backoff gives VLMs a better chance of achieving higher scores. Each time the answer extractor returns an empty answer because the VLMs produced an incoherent answer, we reduce the temperature by 0.1 (starting from 1.0) until it reaches 0.0, at which point the example is skipped.

Models perform slightly better with temperature backoff. We introduced temperature backoff to reduce model incoherence, and it performed fairly well. As shown in Tab. 7 (bottom), this method slightly improves performance across each model, resulting in an average error reduction of 5.78% in CAPTURE^{real} and 5.45% in CAPTURE^{synthetic}. Temperature backoff essentially allows the model to reattempt the question if it fails to respond to the prompt. Similar to previous results, positive results from reattempts highlight VLMs' weak reasoning abilities.

[2]We set $T = 0.9999$ based on manual evaluation, finding it resulted in fewer false positives.

Model	Error (%) [↓]	
	All Objects	Only Occluded
GPT-4o	14.75	26.13 (+11.38)
InternVL2	32.90	75.82 (+42.92)
Molmo	32.49	96.79 (+64.30)
Qwen2VL	29.33	32.89 (+3.56)
Avg. of 4 VLMs	27.37	57.91 (+30.54)

Table 8. VLM sMAPE for counting all objects and counting only the occluded objects in CAPTURE^{real}. Metric: sMAPE (lower is better).

C.3. Models struggle at counting just occluded objects

We separately test whether models can count only the occluded objects (not including the visible objects) in an image. Here, as Tab. 8 demonstrates, the models perform especially poorly in this task, with high error rates across all models. Therefore, we can conclude that occlusion and counting are uniquely difficult for the VLMs, and that the drop in performance between unoccluded and occluded settings in Tab. 2 is likely due to a poor ability to count occluded objects.

C.4. Models are overconfident in occluded settings

We test the uncertainty with two different methods of obtaining confidence on Qwen2VL. In the first method, we prompt Qwen2VL for its confidence in the answer. For the second method, we generate 20 responses for every question in our VQA and calculate the confidence as the percentage of times the most common answer was generated. These results can be seen in Fig. 9 and Fig. 10 respectively. In both reliability curves, there is a slight trend that the model's confidence is negatively correlated with the error, which is the desired outcome. In CAPTURE^{real}, how-

ever, the correlation is much stronger. While the models are somewhat calibrated (with generally lower confidence on higher-error examples, there are still outliers in prompted confidence for CAPTURE^{real} occluded and sampled confidence for CAPTURE^{synthetic} occluded. This indicates that not only do the models perform worse under occlusion, but they can also be overconfident.

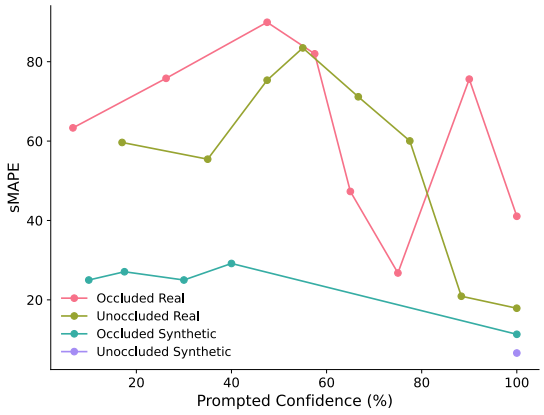

Figure 9. Reliability curve of prompting model for confidence vs. sMAPE.

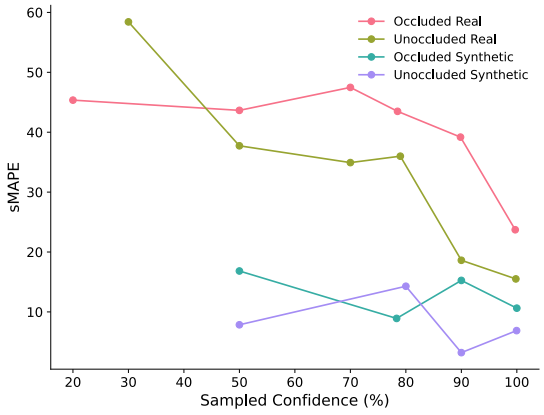

Figure 10. Reliability curve of sampling model for confidence vs. sMAPE.

C.5. Models are biased to predict specific numbers.

To examine where models frequently err, we generated a confusion matrix for every model based on CAPTURE^{synthetic} results (shown in Appendix C.5). The y-axis represents the ground truth values and the x-axis represents the model's answers. We find that models often over-predict numbers associated with common counts in real life: GPT-4o tends to predict numbers like 8, 9, 10, and 12, which are all non-prime numbers (i.e. can be arranged into a grid) and common groupings of objects. For example, 12 is a common grouping (dozens) and allows arrangements into 3x4 or 2x6 grids. InternVL and Qwen2VL over-predict 5 and 10, aligning with how humans conceptualize numbers. Indeed, Coupland [12] found that numbers 5, 10, 20, and other round numbers appear disproportionally more in online texts. Molmo has no correlation with these factors, possibly due to its unique "point and count" ability.

D. VLM Prompts

We use a 100-example validation set for each setting to select the best prompt, which we report below.

Prompt for GPT-4o on CAPTURE^{real} unoccluded split.

Count the exact number of [object] in the image. Assume the pattern of [object] continues behind any black box. Provide the total number of [object] as if the black box were not there.

Prompt for InternVL2 on CAPTURE^{real} unoccluded split.

Your task is to count objects in the image. First, state what the pattern is, then give your final count.

Prompt for Molmo on CAPTURE^{real} unoccluded split.

Count the exact number of [object] in the image. Only count [object] that are visible within the frame. If [object] are partially in the frame (i.e. if any part of [object] are visible), count it.

Prompt for Qwen2VL on CAPTURE^{real} unoccluded split.

Count the exact number of [object] in the image. Assume the pattern of [object] continues behind any black box. Provide the total number of [object] as if the black box were not there. Only count [object] that are visible within the frame (or would be visible without the occluding box). If [object] are partially in the frame (i.e. if any part of [object] are visible), count it. If the [object] would be partially in the frame without the occluding box, count it.

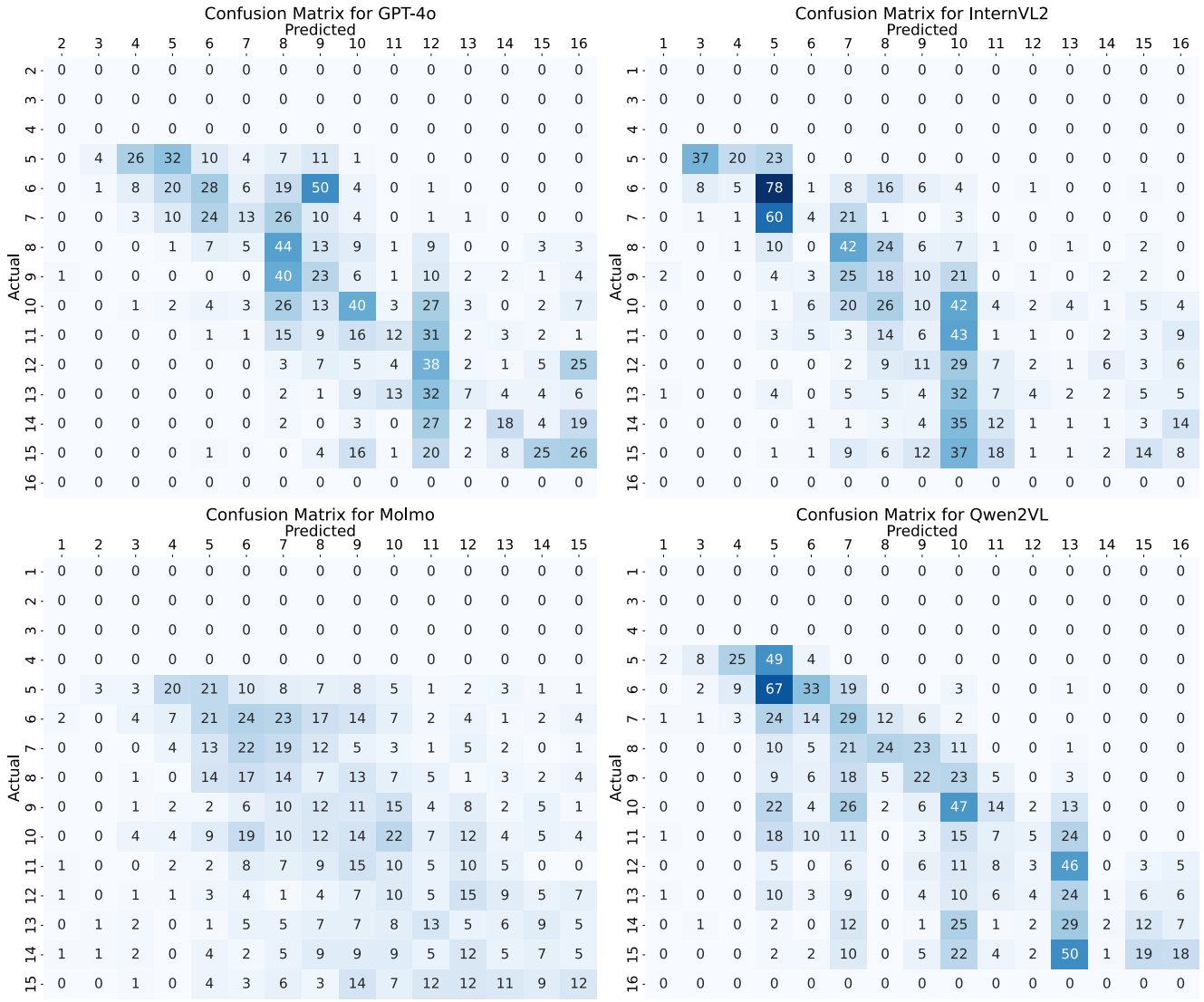

Figure 11. Confusion matrix: predicted vs. ground truth counts for CAPTURE$^{\text{real}}$'s occluded split.

Prompt for GPT-4o, InternVL2, and Qwen2VL on CAPTURE$^{\text{real}}$ occluded split.

Count the exact number of [object] in the image. Assume the pattern of [object] continues behind any black box. Provide the total number of [object] as if the black box were not there. Only count [object] that are visible within the frame (or would be visible without the occluding box). If [object] are partially in the frame (i.e. if any part of [object] are visible), count it. If the [object] would be partially in the frame without the occluding box, count it. Molmo: Your task is to count objects in the image. Assume the pattern of [object] continues behind the black box. First, state what the pattern is, then give your final count.

Prompt for Molmo on CAPTURE$^{\text{real}}$ occluded split.

Your task is to count objects in the image. Assume the pattern of [object] continues behind the black box. First, state what the pattern is, then give your final count.

Prompt for GPT-4o on CAPTURE$^{\text{synthetic}}$ unoccluded split.

Your task is to count objects in the image. First, state what the pattern is, then give your final count.

Prompt for InternVL2 on CAPTURE^{synthetic} unoccluded split.

Count the exact number of [dot shape]s in the image. Only count [dot shape]s that are visible within the frame. If [dot shape]s are partially in the frame (i.e. if any part of [dot shape]s are visible), count it.

Prompt for Molmo on CAPTURE^{synthetic} unoccluded split.

Count the exact number of [dot shape]s in the image. Only count [dot shape]s that are visible within the frame.

Prompt for Qwen2VL on CAPTURE^{synthetic} unoccluded split.

Count the exact number of [dot shape]s in the image. Assume the pattern of [dot shape]s continues behind any black box. Provide the total number of [dot shape]s as if the black box were not there. Only count [dot shape]s that are visible within the frame (or would be visible without the occluding box). If [dot shape]s are partially in the frame (i.e. if any part of [dot shape]s are visible), count it. If the [dot shape]s would be partially in the frame without the occluding box, count it.

Prompt for GPT-4o and Molmo on CAPTURE^{synthetic} occluded split.

Your task is to count objects in the image. Assume the pattern of [dot shape]s continues behind the black box. First, state what the pattern is, then give your final count.

Prompt for InternVL2 and Qwen2VL on CAPTURE^{synthetic} occluded split.

Count the exact number of [dot shape]s in the image. Assume the pattern of [dot shape]s continues behind any black box. Provide the total number of [dot shape]s as if the black box were not there. Only count [dot shape]s that are visible within the frame (or would be visible without the occluding box). If [dot shape]s are partially in the frame (i.e. if any part of [dot shape]s are visible), count it. If the [dot shape]s would be partially in the frame without the occluding box, count it.