Watch And Learn: Empowering MLLMs to Count Like Humans

Qixin Xu
Department of Computer Science and Technology
Tsinghua University
xqx23@mails.tsinghua.edu.cn

Siqiao Huang
Institute for Interdisciplinary Information Sciences
Tsinghua University
huang-sq23@mails.tsinghua.edu.cn

Yiming Liu
Institute for Interdisciplinary Information Sciences
Tsinghua University
liuym23@mails.tsinghua.edu.cn

Abstract

Vision-Language Models (VLMs) face challenges in performing accurate counting, a critical task requiring both visual perception and reasoning. Current VLMs separate these processes, limiting their ability to handle counting tasks effectively. To address this, we propose the "Watch-and-Learn" framework, which enables continuous visual perception during counting by integrating two function calls, “Point Annotation” and “See Image.” This approach allows the model to iteratively process visual information, improving its reasoning ability. Additionally, we use Reinforcement Learning Algorithms to further refine the model’s performance in an online learning environment. Our method enhances VLMs’ counting capabilities, bridging the gap between visual perception and reasoning.

1 Introduction

In recent years, Vision-Language Models (VLMs) have made significant strides, powered by the vast world knowledge encoded in Large Language Models (LLMs) [1, 3, 19, 9]. These multimodal models have shown impressive capabilities in tasks such as visual question answering [11], visual grounding [15], and optical character recognition [21], demonstrating their potential in handling diverse visual and textual inputs. The typical training pipeline for VLMs consists of two key phases [4, 17, 20]: first, developing a robust visual understanding through exposure to large datasets of image-caption pairs, and second, fine-tuning the model to enable problem-solving based on these visual inputs. While this process has yielded substantial improvements in a wide array of tasks, there remains a critical limitation—VLMs often fail to engage in the type of intermediate visual reasoning required for tasks that demand detailed and nuanced understanding.

A notable example of such a limitation arises in counting, an operation seemingly simple for humans but surprisingly challenging for current VLMs. Humans intuitively approach counting as an iterative process, revisiting and refining their attention to visual scenes by focusing on specific objects or regions multiple times. This cyclical approach—marked by a continuous back-and-forth between

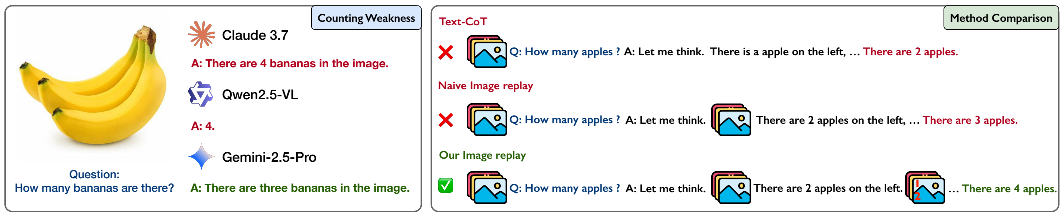

Counting Weakness
Claude 3.7
A: There are 4 bananas in the image.
Qwen2.5-VL
A: 4.
Gemini-2.5-Pro
Question:
How many bananas are there?
A: There are three bananas in the image.
Method Comparison
Text-CoT
Q: How many apples ? A: Let me think. There is a apple on the left, … There are 2 apples.
Naive Image replay
Q: How many apples ? A: Let me think. There are 2 apples on the left, … There are 3 apples.
Our Image replay
Q: How many apples ? A: Let me think. There are 2 apples on the left. … There are 4 apples.

Figure 1: Comparision between Text-CoT, Image Replay without annotation and Our Image Replay Method.

perception and reasoning—is a key element of human cognition. In contrast, VLMs process visual inputs in a single pass, separating perception from reasoning and leading to a loss of the necessary intermediate steps for tasks like counting. The one-time pass through the image means the model is unable to dynamically adjust its focus or reasoning based on evolving visual information, ultimately hindering its performance in tasks that require this kind of iterative process. While significant progress has been made in improving VLMs' overall capabilities, their performance in counting tasks and other tasks that demand fine-grained visual attention continues to fall short. This gap highlights the need for new approaches that integrate continuous visual perception with logical reasoning, enabling VLMs to better mimic human cognitive processes. Inspired by how humans tackle complex visual problems through manipulations—such as zooming in on specific regions, marking points of interest, or revisiting parts of an image—we propose a novel method to address this gap. By enabling VLMs to perform counting tasks iteratively, using continuous visual attention and dynamic reasoning, we aim to bridge the gap between visual perception and reasoning, improving model accuracy and reliability in handling tasks that involve detailed visual analysis.

Our work introduces the "Watch-and-Learn" framework, a solution that allows VLMs to continuously perceive and reason about visual inputs during tasks like counting. This approach integrates the model's ability to focus on different regions of an image through iterative perception, ensuring that each step of the counting process builds on the previous one. By adopting this method, we demonstrate a significant improvement in the model's ability to handle visual reasoning tasks across four counting benchmarks, providing new insights into how VLMs can be better trained to address challenges in real-world applications requiring precise and iterative visual reasoning.

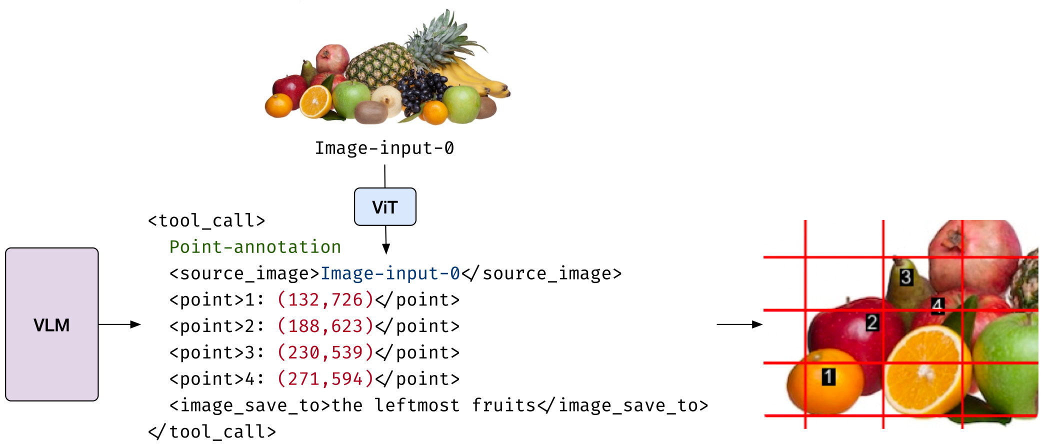
2 Terminology

Counting, a task that is inherently simple for humans, presents a significant challenge for Vision-Language Models (VLMs). The difficulty lies in the fact that counting requires two essential capabilities: visual perception and visual reasoning [7].

Visual perception refers to the model's ability to identify and localize relevant objects within an image, allowing it to determine where to direct its attention. This foundational skill enables the model to distinguish between different elements in the visual scene. On the other hand, visual reasoning pertains to the model's ability to construct logical thought trajectories based on the visual context provided by the perception module. For example, using a Vision Transformer (ViT), the model processes the visual input and interprets the observed scene, building a coherent understanding necessary for complex reasoning tasks.

The challenges VLMs face when performing counting tasks can be attributed to two main factors: the lack of continuous visual perception and the disjointed relationship between visual perception and reasoning during training. In human counting, individuals engage in an iterative, cyclical process—observing, counting, verifying, and then recounting—focusing repeatedly on different regions of the image to refine their understanding. In contrast, VLMs typically only view an image once, where the visual tokens are concatenated alongside the textual tokens in a single input sequence. This static process limits their ability to re-focus and adjust attention based on evolving insights, creating a disconnect between the initial perception and the reasoning that follows.

Moreover, while human counting is inherently a dynamic process, where visual perception and reasoning continuously interact, VLMs are trained in a manner that separates these processes. The



```
<tool_call>
  Point-annotation
  <source_image>Image-input-0</source_image>
  <point>1: (132,726)</point>
  <point>2: (188,623)</point>
  <point>3: (230,539)</point>
  <point>4: (271,594)</point>
  <image_save_to>the leftmost fruits</image_save_to>
</tool_call>
```

Figure 2: Point Annotation retrieves an image from the buffer along with a set of coordinates generated by the model, labels the image according to these coordinates, and subsequently saves the annotated image back to the buffer.

perception is fixed in the prompt, and reasoning is separated by the model's output. This division results in a lack of continuity during tasks that demand attention to detail and refinement across multiple steps.

To address these issues, we introduce a novel framework called "Watch-and-Learn". In this framework, we propose two function calls designed to facilitate a continuous process of visual perception and reasoning: Point Annotation and See Image. With these two function call, the whole counting process can be formulated as Algorithm 1.

Point Annotation takes an image and a set of coordinates as input and returns an image with labeled annotations based on those coordinates. During the counting process, the model will output the coordinates of the objects it counts. This function call receives an image and a set of coordinates representing the identified objects, then marks these coordinates on the image with labels or markers. During training, we found that the coordinates generated by the policy model may not be accurate, so we add grid and axis to strengthen model's marking accuracy. The annotated image is stored in an image buffer, allowing it to be preserved for future reference. This image buffer holds the modified image until the See Image function is invoked, at which point the model can access the labeled image tokens from the buffer to continue processing and refining the counting task.

See Image takes an image as input and outputs a series of image tokens corresponding to that input. Specifically, the function retrieves a previously annotated image from the buffer, which has been marked with the coordinates of the counted objects. This annotated image is then encoded through a Vision Transformer (ViT) into a sequence of image tokens. These image tokens are appended to the current decoding sequence, allowing the model to integrate the visual information dynamically. As the model continues decoding, the newly added image tokens become part of the attention mechanism, enabling the model to focus on the visual details within the image. This process ensures that the model can continuously attend to and incorporate new visual information throughout the counting process, fostering a seamless interaction between perception and reasoning.

3 Experiments

3.1 Experimental Setup

Models. For the purpose of our experiments, we utilize `Qwen2.5-VL-Instruct-7B` [5], a state-of-the-art vision language model. This model is part of the Qwen2.5 series and is designed to handle

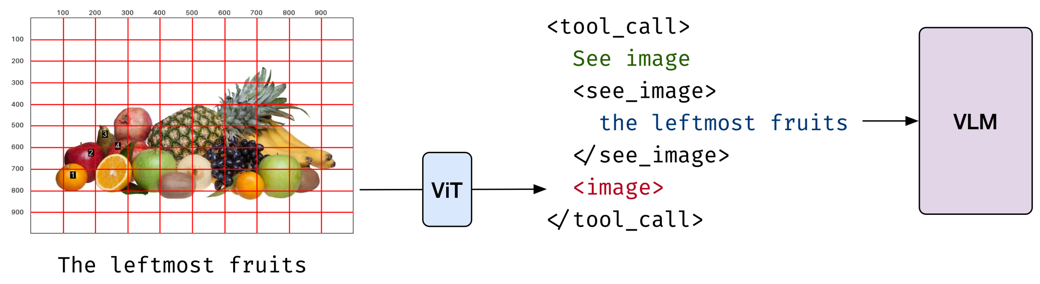


Figure 3: When calling "See Image", the <image> token will be replaced with the encoding of the annotated image from ViT.

Algorithm 1 Counting Process using "Watch-and-Learn" Framework

```
1: Input: Image $I$
2: Initialize Image Buffer $B$
3: Initialize Token Sequence $T$
4: Initialize Counter $count = 0$
5: while counting is not finished do
6:     Generate Next Token: Decode next token from sequence $T$    ▷ Model continues decoding sequence
7:     if next token is Point Annotation then
8:         Point Annotation(Image $I$) ▷ Model calls Point Annotation to mark points on the image
9:         Annotated Image $I_{annot}$ $\leftarrow$ Store annotated image in buffer $B$
10:        Increment $count$                                    ▷ Update the counting result
11:    else if next token is See Image then
12:        See Image(Image $I_{annot}$) ▷ Model retrieves annotated image from buffer and encodes it
13:        Image Tokens $T_{image} \leftarrow$ ViT($I_{annot}$)     ▷ Encode annotated image into token sequence
14:        Append $T_{image}$ to $T$    ▷ Add image tokens to the current sequence for future decoding
15:    end if
16: end while
17: Output: Final Count $count$
```

multimodal tasks involving both visual and textual inputs. We choose the parameter size of 7B for balancing performance with training costs.

Training Data. For Supervised Finetuning, we utilize a subsection of 20k from `PixMo-Points` [10], a dataset of images paired with referring expressions and points marking the locations the referring expression refers to in the image. The dataset contains a diverse range of points and expressions, with many high-frequency (10+) expressions annotated by human annotators. For Reinforcement Learning Finetuning [13], we utilize a 10k subset of `PixMo-Points`, detailed data processing pipelines can be seen Section 3.2.

Task Description. Counting encompasses two key dimensions: Object Counting and Instructional Counting. Object Counting involves the simple task of counting objects in an image, requiring basic visual perception to identify and enumerate instances of the objects. Models like Grounding DINO, which specialize in object detection and localization, are particularly effective in this domain. On the other hand, Instructional Counting extends this task by adding specific conditions or instructions, such as counting only those objects that meet certain attributes (e.g., "How many people are wearing red hats?"). This requires both perceptual reasoning and the ability to follow instructions, making it more complex than Object Counting. While Grounding DINO excels at the former, its ability to handle the conditional reasoning in Instructional Counting remains limited. Both dimensions are essential for a comprehensive evaluation of counting tasks, measuring the model's ability to handle both basic object recognition and more complex, instruction-driven reasoning.

Model	Instructional Counting				Object Counting				AVG Acc.	AVG RMSE
	Pixmo-point 500 [10]		Counting Bench		FSC-147 [16]		Pixmo Count Test [10]			
	Acc↑	RMSE↓	Acc↑	RMSE↓	Acc↑	RMSE↓	Acc↑	RMSE↓		
GPT-4o [12]	30.9	8.3	85.4	1.1	9.7	103.2	50.8	2.1	20.3	55.8
Gemini 2.5 Flash [8]	38.4	16.2	83.2	1.1	12.0	334.0	68.8	1.4	25.2	175.1
Qwen2.5 VL 72B [6]	39.2	7.6	83.7	1.2	12.7	64.9	63.5	1.5	26.0	36.3
Qwen2.5 VL 7B [6]	25.2	15.2	80.7	1.0	7.7	118.1	51.6	1.8	16.5	66.7
CountGD [2]	17.6	62.8	75.4	8.3	28.7	119.4	70.9	8.5	23.2	91.1
SFT	44.2	7.7	73.4	2.6	14.7	152.4	67.8	1.3	29.5	80.1
Replay RL	44.1	6.2	74.2	3.2	18.6	139.8	71.6	1.6	31.4	73.0

Table 1: A performance comparison of various foundation models on instructional and object counting tasks, evaluated by accuracy (Acc↑) and Root Mean Square Error (RMSE↓) across multiple benchmarks.

3.2 Data Processing

The PixMo-Points dataset consists of images containing various objects, along with their corresponding coordinates. In the data processing pipeline, we first apply the DBSCAN [18] clustering algorithm to partition the set of coordinates $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ into distinct groups, denoted as clusters $\mathcal{C} = \{C_1, C_2, ..., C_m\}$, where each $C_i \subset \mathcal{P}$ is a subset of points in the i-th cluster.

Subsequently, within each cluster C_i, we sort the points $\{p_j \in C_i\}$ according to their position in a direction from the upper-left to the lower-right. This sorting is based on a predefined metric $f(p_j)$ that compares the positions of the points, such as their coordinates (x_j, y_j), so that for all $p_j, p_k \in C_i$:

$$f(p_j) < f(p_k) \quad \iff \quad (x_j, y_j) \preceq (x_k, y_k)$$

Finally, we arrange the clusters themselves, $\mathcal{C}$, in the same directional order, ensuring consistency across the dataset. This ordering is also determined by the positions of the centroids c_i of each cluster C_i, where the centroid c_i is computed as:

$$c_i = \left(\frac{1}{|C_i|} \sum_{p_j \in C_i} x_j, \frac{1}{|C_i|} \sum_{p_j \in C_i} y_j \right)$$

Thus, the clusters are sorted such that:

$$c_i \preceq c_k \quad \text{for} \quad i < k$$

Additionally, we applied other counting strategies to the raw dataset, resulting in different supervised fine-tuning data. The details of these strategies can be found in Section 3.5.2.

3.3 Evaluation.

Benchmarks. For the task of Instruction Counting, we validate the effectiveness of our approach under two well-established benchmarks:

(i). **Pixmo-Point** [10], in-domain data with diverse range of point annotations.

(ii). **CountBench** [14], a image-text counting benchmark for evaluating a model's understanding of object counting.

For the task of Object Counting, we utilize two benchmarks:

(i). **FSC147** [16], a dataset of 147 object categories containing over 6000 images that are suitable for the few-shot counting task.

(ii). **Pixmo-Count** [10], a dataset of images paired with objects and their point locations in the image.

Model	Instructional Counting				Object Counting				AVG Acc.	AVG RMSE
	Pixmo-point 500 [10]		Counting Bench		FSC-147 [16]		Pixmo Count Test [10]			
	Acc↑	RMSE↓	Acc↑	RMSE↓	Acc↑	RMSE↓	Acc↑	RMSE↓		
Text-CoT RL (step=0)	44.1	7.2	75.4	2.6	15.8	153.8	66.5	1.4	30.0	80.5
Text-CoT RL (step=280)	41.4	6.9	71.9	2.3	13.1	138.7	65.4	1.2	27.3	72.8
Image Replay RL (step=0)	44.2	7.7	73.4	2.6	14.7	152.4	67.8	1.3	29.5	80.1
Image Replay RL (step=80)	44.1	6.2	74.2	3.2	18.6	139.8	71.6	1.6	31.4	73.0

Table 2: Performance of Text-CoT and Image Replay with Reinforcement Learning Finetuning. Step indicates the number of gradient steps in RL finetuning, where step=0 represents the original model with only the incorporation of the proposed technique.

Evaluation Metrics. To validate our model's performance in counting tasks, we adopt the following two metrics:

(i). **Accuracy (Acc):** This metric measures the proportion of examples where the predicted count exactly matches the ground truth. It reflects the model's ability to produce precise predictions and is defined as:

$$Acc = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[\hat{y}_i = y_i], \tag{1}$$

where $\hat{y}_i$ is the predicted count, y_i is the ground-truth count, and $\mathbb{I}[\cdot]$ is the indicator function.

(ii). **Root Mean Square Error (RMSE):** This metric evaluates the overall deviation between predicted and ground-truth counts. It penalizes large errors more severely and is computed as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}. \tag{2}$$

RMSE provides a measure of the model's average prediction error magnitude across the dataset.

3.4 Main Results

3.5 Analysis

In this section, we analyze the benefits of Image Replay compared to Text-CoT, as well as the influence of different Counting Strategies.

3.5.1 Image Replay vs. Text-CoT

We compare the performance of using Image Replay against the baseline method of Text-CoT for Reinforcement Learning fine-tuning. As shown in Table 3.5.1, simply adopting reinforcement learning with Text-CoT on counting tasks leads to a performance drop, whereas Image Replay RL continues to improve. We deduce that this is due to our replay algorithm, which simultaneously trains the perception and reasoning modules which is important for counting tasks. Notably, Image Replay RL achieves superior performance on counting tasks with fewer gradient steps, demonstrating the effectiveness and sample efficiency of our proposed Image Replay method.

3.5.2 Different Counting Strategies

In this section, we explore different strategies of doing point annotations for the given image. Specifically we consider three strategies:

- **Random Ordering**: The ordering of the objects is assigned randomly, agnostic to the underlying clusters.
- **Cluster Ordering**: The ordering of the objects is assigned based on Clusters. Inside each cluster, the ordering is assigned randomly.

Model	Instructional Counting				Object Counting				AVG Acc.	AVG RMSE
	Pixmo-point 500 [10]		Counting Bench		FSC-147 [16]		Pixmo Count Test [10]			
	Acc↑	RMSE↓	Acc↑	RMSE↓	Acc↑	RMSE↓	Acc↑	RMSE↓		
Image Replay w/o Annotation	40.2	8.1	-	-	-	-	-	-	-	-
Image Replay + Random Ordering	38.0	8.2	66.0	2.8	12.0	153.6	65.2	0.7	1.5	-
Image Replay + Cluster Ordering	44.2	7.7	73.4	2.6	14.7	152.4	67.8	1.3	29.5	80.1
Image Replay + Direction Ordering	44.3	7.5	82.7	2.6	21.0	156.2	74.3	2.5	32.7	81.9

Table 3: Performance of Image Replay with different counting strategies.

- **Direction Ordering**:The ordering of the objects is assigned based on Clusters. Inside each cluster, the ordering is assigned based on directions.

These three strategies resemble the strengthening of deterministic ordering in a progressive manner, i.e. Random Ordering forces no ordering constraints at all, while Cluster Ordering enforces the ordering of clusters, and Direction Ordering take this a step further by defining the ordering inside each cluster.

As shown in Table 3.5.2, while all annotation strategies except Random Ordering outperform image replay with no annotations at all, the annotation strategies significantly impact the model's performance. Namely, the overall performance of Image Replay with Direction Ordering is the strongest, followed by Image Replay with Cluster Ordering, and Random Ordering achieves the worst performance.

3.6 Case Study

In this section, we present two inference results from our model, which has been trained through both the Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) phases. The first result shown in 3.6 represents a scenario in which the objects are well-aligned, while the second result shown in 3.6 reflects a scenario where the objects being counted are intricately arranged. Regarding the marking tags appearing in the image, objects with the same tag are grouped into a single cluster (i.e., counted within one function call), and the tag count increases accordingly, e.g. the objects labelled as "1" are counted first, then objects labelled as "2" are counted.

4 Conclusion

In this paper, we introduced the "Watch-and-Learn" framework to enhance the counting capabilities of Vision-Language Models (VLMs). By integrating continuous visual perception through iterative processes, our approach enables the model to dynamically adjust its focus and reasoning, closely mimicking human cognitive strategies for complex visual tasks. This method bridges the gap between visual perception and reasoning, significantly improving VLMs' performance in tasks that demand detailed and iterative visual analysis. We demonstrated the effectiveness of our framework across four counting benchmarks, showing that our approach not only enhances the accuracy and reliability of VLMs in counting tasks but also lays the foundation for more advanced solutions in other domains requiring similar fine-grained visual attention. The use of Reinforcement Finetuning further refines the model's performance, allowing it to adapt in an online learning environment. Our work provides valuable insights into how VLMs can be better trained to handle real-world challenges, offering a new direction for improving multimodal models' reasoning capabilities.

Acknowledgments and Disclosure of Funding

The authors wish to convey their deepest gratitude to Professor Hang Zhao for his exceptional guidance and unwavering support throughout the course of this research. They would also like to extend special thanks to Dr. Baijun Ye for his invaluable insights and substantial contributions to the advancement of this work.

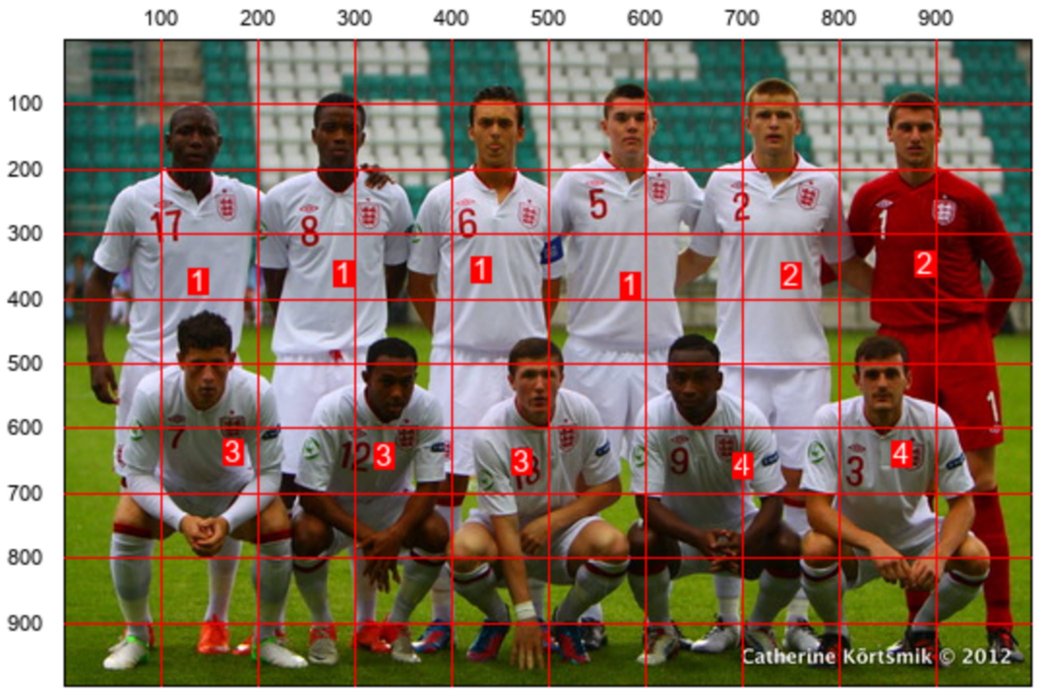


Figure 4: The well-aligned counting scenario, our model counts row by row in horizontal order.

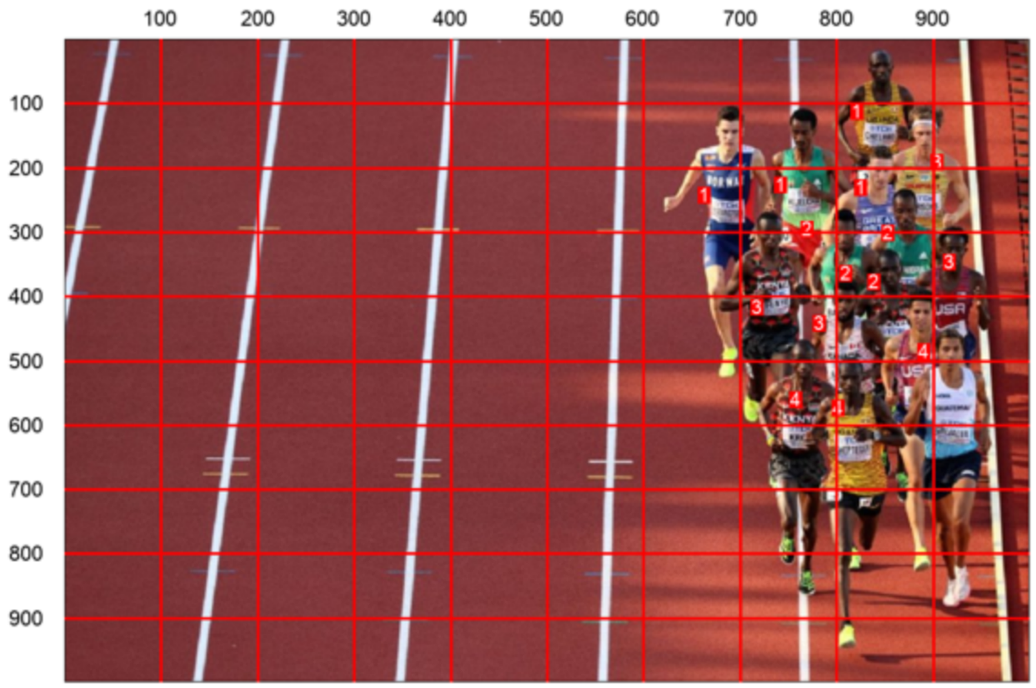


Figure 5: The intricately arranged counting scenario, our model counts cluster by cluster from upper-left to lower-right.

References

[1] Manoj Acharya, Kushal Kafle, and Christopher Kanan. Tallyqa: Answering complex counting questions, 2018. 1

[2] Niki Amini-Naieni, Tengda Han, and Andrew Zisserman. Countgd: Multi-modal open-world counting. *Advances in Neural Information Processing Systems*, 37:48810–48837, 2024. 5

[3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023. 1

[4] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023. 1

[5] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv: 2502.13923*, 2025. 3

[6] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 5

[7] Felix Chen, Hangjie Yuan, Yunqiu Xu, Tao Feng, Jun Cen, Pengwei Liu, Zeying Huang, and Yi Yang. Mathflow: Enhancing the perceptual flow of mllms for visual mathematical problems, 2025. 2

[8] Google Cloud. Gemini 2.5 flash | generative ai on vertex ai | google cloud, 2025. Accessed 2025-06-13. 5

[9] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023. 1

[10] 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, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. *Cvpr*, 2025. 4, 5, 6, 7

[11] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. 1

[12] Openai.com. Hello gpt-4o, 2025. Accessed 2025-06-13. 5

[13] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv: 2203.02155*, 2022. 4

[14] Roni Paiss, Ariel Ephrat, Omer Tov, Shiran Zada, Inbar Mosseri, Michal Irani, and Tali Dekel. Teaching clip to count to ten. *arXiv preprint arXiv: 2302.12066*, 2023. 5

[15] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world, 2023. 1

[16] Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 5, 6, 7

[17] Quan Sun, Qiying Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, Yueze Wang, Hongcheng Gao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Emu: Generative pretraining in multimodality, 2024. 1

[18] Daren Wang, Xinyang Lu, and Alessandro Rinaldo. Dbscan: Optimal rates for density based clustering, 2019. 5

[19] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. Cogvlm: Visual expert for pretrained language models, 2024. 1

[20] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm-v: A gpt-4v level mllm on your phone, 2024. 1

[21] Yanzhe Zhang, Ruiyi Zhang, Jiuxiang Gu, Yufan Zhou, Nedim Lipka, Diyi Yang, and Tong Sun. Llavar: Enhanced visual instruction tuning for text-rich image understanding, 2024. 1