

# Multimodal LLMs Do Not Compose Skills Optimally Across Modalities

Paula Ontalvilla, Aitor Ormazabal, Gorka Azkune

HITZ Center - Ixa, University of the Basque Country (EHU)  
{paula.ontalvilla, aitor.ormazabal, gorka.azkune}@ehu.eus

## Abstract

Skill composition is the ability to combine previously learned skills to solve new tasks. As neural networks acquire increasingly complex skills during their pretraining, it is not clear how successfully they can compose them. In this paper, we focus on Multimodal Large Language Models (MLLM), and study their ability to compose skills across modalities. To this end, we design three evaluation tasks which can be solved sequentially composing two modality-dependent skills, and evaluate several open MLLMs under two main settings: i) prompting the model to directly solve the task, and ii) using a two-step cascaded inference approach, which manually enforces the composition of the two skills for a given task. Even with these straightforward compositions, we find that all evaluated MLLMs exhibit a significant **cross-modality skill composition gap**. To mitigate the aforementioned gap, we explore two alternatives: i) use chain-of-thought prompting to explicitly instruct MLLMs for skill composition and ii) a specific fine-tuning recipe to promote skill composition. Although those strategies improve model performance, they still exhibit significant skill composition gaps, suggesting that more research is needed to improve cross-modal skill composition in MLLMs.

**Keywords:** Evaluation Methodologies, Training, Fine-tuning, Adaptation, Alignment, and Representation Learning, Skill Composition, Cross-modality, Multimodality

## 1. Introduction

Multimodal Large Language Models (MLLM) have become increasingly capable in recent years, receiving ever-growing interest from the research community and industry (Alayrac et al., 2022; Driess et al., 2023; Liu et al., 2023; OpenAI et al., 2024b). However, even as these models become capable of solving more and more complex tasks, the mechanisms through which they do so remain unclear, spawning extensive research into their internal workings (Dang et al., 2024).

MLLMs are often trained using a capable LLM as a starting point, and visual or other multimodal capabilities are added by further fine-tuning it on modality-specific tasks. While some work has studied the effects of this type of training on the original model’s textual capabilities (Lin et al., 2023; Dai et al., 2024), the resulting MLLM’s ability to coherently combine its visual and textual skills is not well understood. In this work, we pose the following question: *Can MLLMs successfully compose skills across modalities?*

More specifically, we devise 3 image-to-text tasks that can be solved by composing a visual skill with a textual one, in a way that would be trivial for a human: 1) Reasoning over questions rendered in an image (OCR + textual reasoning), 2) Object counting (object identification + counting), and 3) Card playing (card identification + textual reasoning over the game state). By comparing the model’s natural performance on these tasks with a cascaded inference approach that forces skill composition by

making two calls to the same model, we can measure a “cross-modality skill composition gap”. We measure this gap on a wide array of popular open-source MLLMs and datasets and find that, strikingly, **all tested models exhibit a significant cross-modality skill composition gap**, even when the required combination is straightforward<sup>1</sup>.

Having demonstrated the existence of this gap, we explore two possible mitigation strategies: one leveraging a prompt specifically constructed to induce the relevant skill composition and a second approach exploring fine-tuning to induce cross-modal skill composition. We show that the specially constructed prompt partially reduces the skill composition gap, but it suffers from lack of scalability as it requires a task-specific prompt. Fine-tuning also improves the results and it contributes to generalization to some extent, yet the gap remains.

Our results show that MLLMs do not optimally combine their visual and textual skills even in very straightforward scenarios, prompting further research into the root cause and effective mitigation of this phenomenon.

## 2. Related Work

### 2.1. Skill composition in neural networks

There are different ways to understand and approach **skills** and **skill composition** in the liter-

<sup>1</sup><https://github.com/paulaonta/skillComposition>

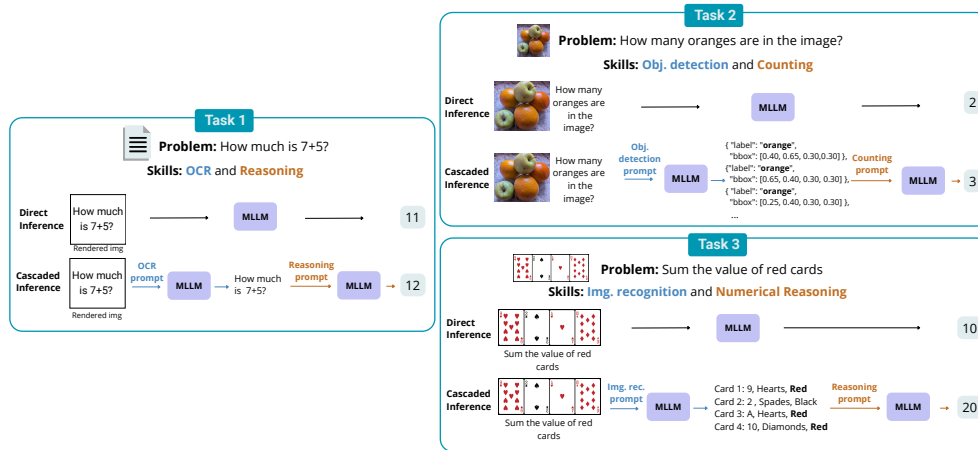


Figure 1: We define three tasks that can be solved with a trivial sequential composition of two modality-specific skills. We evaluate MLLMs on those tasks using direct inference, i.e. standard inference. We compare the performance with the cascaded inference, where we manually induce the required skill composition with two calls to the MLLM.

ature, both in robotics (Qureshi et al.; Liu et al., 2025) and in LLMs (Zhao et al., 2024). We will cover the literature of the domains that are closest to our work.

In the artificial agent community, skills are defined as separated atomic tools that an agent can use at inference time via API calls (Zheng et al., 2025; Lu et al., 2023). Thus, skill composition is the ability of those agents to compose the tools available to solve different tasks. To achieve skill composition Zheng et al. (2025) focus on self-improvement techniques proposing a structured pipeline that allows web agents to autonomously discover, refine, and recombine reusable skills over time. In contrast, Lu et al. (2023) rely on an LLM to plan a sequence of tools to be used to approach a given task.

The LLM community offers some other views on skill composition. Zhao et al. (2024) defines compositional generalization as “the capacity to combine learned skills in novel ways that are not encountered during training”. We adopt this definition in our formulation of skill composition. They used the SkillMix dataset (Yu et al., 2023) to show that LLMs can learn to compose skills to some extent through specific fine-tuning. In this framework, the skills they consider are *literary, rhetorical, reasoning, logic, theory of mind, pragmatics, common sense and physical knowledge*.

We also understand skills as specific abilities learned by neural networks during training. However, rather than focusing on diffuse skill categories and complex compositions for text-only LLMs as in Yu et al. (2023), we focus on the multimodal scenario, where visual and textual skills can be sharply divided, and study straightforward scenarios that can be solved by combining one **visual** skill

with one **textual** skill. This allows us to measure the cross-modality skill composition gap, and study how well MLLMs are able to compose skills across modalities.

## 2.2. Multimodal Large Language Models (MLLM)

In this paper, we will focus on Multimodal Large Language Models (MLLMs) that receive visual and textual input and generate textual output. Following the current dominant paradigm, we consider open weights models that follow the late fusion paradigm, where a pretrained visual encoder and LLM are connected and further trained for vision-language instruction following. Examples of this paradigm are the LLaVA family (Liu et al., 2024a), Llama 3.2 (Grattafiori et al., 2024), Molmo (Deitke et al., 2024) and Qwen2.5-VL (Bai et al., 2025).

Studying skill composition requires visibility into when and how individual skills are acquired during training. For this reason, we restrict our analysis to open-source MLLMs, whose training stages and objectives are publicly documented. Closed-source models are excluded, as their training data and process are not transparent, making it impossible to determine which skills they were explicitly trained for.

Open-source MLLMs offer an ideal test bench to study skill composition for two reasons. Firstly, modality-specific skills can be easily distinguished and tracked on their training stages. For example, current MLLMs are often trained to perform Optical Character Recognition (OCR), which we categorize as a visual skill (see §3). In contrast, modern pretrained text-only LLMs are commonly trained on data that supports arithmetic reasoning, a textual skill. Secondly, the dominant late-fusion

paradigm adds a modality specific (e.g. visual) encoder later in training, raising questions of how well the final model is able to combine skills from both modalities.

### 3. Skill Composition and Multimodal Large Language Models

In this section, we describe how we understand skills and skill composition, and how we design our tasks to study cross-modality skill composition ability in MLLMs.

A **skill** is the ability acquired by a model as a result of the training process. For instance, a model trained to describe images will acquire the skill of image captioning. In this paper, we distinguish between visual skills, that demand image processing and representation, and textual skills, that only operate on text.

A **task** is a set of instructions and correct answers that are used to measure specific skills. Tasks and skills are closely related, since tasks are used to guide the MLLM training process. In the example before, if we use the task of image captioning to train an MLLM, the model will acquire the image captioning skill. However, the relation between skills and tasks is not always 1 : 1, but  $N : 1$ , meaning that, generally, a set of skills may be needed to solve a task. In that sense, we can define tasks that are not used in the training process but can be solved using the skills acquired in that training process.

Based on these definitions, we will adhere to the definition of compositional generation of (Zhao et al., 2024) and state that: **skill composition** is the capacity to combine learned skills in novel ways that are not encountered during training.

In the context of MLLMs, the pre-trained backbone LLM will have acquired a set of textual skills, while the multimodal fine-tuning will imbue the model with visual skills. We aim to study whether trained MLLMs can successfully combine and compose these skills across modalities. To that end, we design three tasks that can be solved by straightforwardly composing one visual skill and one textual skill, as can be seen in Figure 1.

**Task 1: Reasoning over rendered text.** Given a textual problem, which is rendered as an image, the model is asked (in textual form) to solve the problem which requires some sort of reasoning: mathematical, general knowledge or commonsense reasoning. Task 1 can be solved composing Optical Character Recognition (OCR), a visual skill, and language-based reasoning, a textual skill.

**Task 2: Object counting.** Given an image, the model is asked by text to count the occurrences of a

specific object. This task can be solved composing the visual skill of object detection and the textual skill of counting a given string in a text.

**Task 3: Card playing.** Given an image containing four cards from a poker deck, we instruct the model to order cards or to perform summations under different conditions. To solve those tasks, the visual skill of image recognition and the textual skill of numerical reasoning can be composed.

## 4. Experimental Settings

In this section, we describe the models we chose for our experimentation (§4.1), as well as our approach for measuring the skill composition gap (§4.2).

### 4.1. Models

We evaluate a range of popular open MLLMs with known training skills and tasks, specifically LLaVA 1.6-Vicuna-13B, LLaVA 1.6-Mistral-7B (Liu et al., 2024b), Llama 3.2 11B (Grattafiori et al., 2024), Molmo-7B and 72B (Deitke et al., 2024) and Qwen2.5-VL-7B and 72B (Bai et al., 2025). We do not evaluate closed models, as their training recipes and their backbone LLMs are unknown, and consequently, we cannot identify the specific visual and textual skills they have been explicitly trained for. Based on the papers and technical reports corresponding to each chosen model, we verify that all the MLLMs we selected have the visual and textual skills required by the tasks described in §3, except for LLaVA and Llama models, which have not been explicitly trained for object detection.

### 4.2. Measuring the skill composition gap

Our goal is to evaluate how well MLLMs can compose their visual and textual skills. To that end, we evaluate models on two different inference setups: (1) *Direct inference*, where the model is prompted with an image and an associated question to directly solve the task; (2) *Cascaded inference* which enforces skill composition by making two calls to the same model. Direct inference will show the natural behavior of the MLLMs, letting them solve the task without further instructions. In cascaded inference, on the other hand, we first ask the MLLM to use the relevant visual skill for the target task to produce an intermediate result. This result is then used as the input of the second step, where the MLLM is explicitly asked to use the relevant textual skill (see Figure 1).

Based on these two setups, we define the **cross-modality skill composition gap**<sup>2</sup> of an MLLM as

<sup>2</sup>For simplicity, we use *cross-modality skill composition gap* and *skill composition gap* interchangeably

the difference of performance between cascaded and direct inferences. If the model’s performance under direct inference is lower than the cascaded version, we can conclude that the model is not composing its own skills in an optimal way, since cascaded inference only uses the model itself to solve the individual skills.

Finally, as an upper bound for cascaded inference, we also provide the oracle results when possible. The `oracle_inference` prompts the models to use the relevant textual skill for the target task, but using as input the gold standard result of the visual skill<sup>3</sup>. Thus, the oracle inference setup provides the best results an MLLM could achieve with our designed skill composition.

When evaluating model generations, we distinguish between two cases: i) multiple-choice answers, where models output answers in letter forms and we compute exact-match accuracy; ii) open-ended generation, depending on the dataset, evaluated via regex or LLM-based methods as standard (see Appendix A). For both direct and cascaded inference approaches, we prompt the model to reason step-by-step. See Appendix B for the exact prompts used in each task and setup.

## 5. Results

### 5.1. Results for Task 1: Reasoning over rendered text

As mentioned in §3, we design Task 1 to be solvable by composing the visual skill of Optical Character Recognition (OCR) with the textual skill of reasoning. Since we use open models, we know that all of them have been trained for both OCR and language-based reasoning. To build relevant datasets for Task 1, we render common text-only benchmarks which assess mathematical reasoning (GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2024), using the test set from Lightman et al. (2023)), common-sense reasoning (ARC Easy and Challenge (Clark et al., 2018)) and general knowledge and reasoning ability (MMLU (Hendrycks et al., 2020)). For all datasets, we render the plain text as images. An exception is MATH, which contains LaTeX-formatted content; so, we first compile it before rendering. Since we construct the visual dataset from an original textual one, we naturally have access to the gold results for the OCR task (visual skill) and we can provide results for oracle inference.

---

throughout the paper.

<sup>3</sup>For instance, for Task 1, *reasoning over rendered text*, oracle would prompt the model with the full original textual problem statement, instead of using the model’s own OCR result.

Table 1 reports our results across all datasets under the three inference setups. As shown in the table **all tested MLLMs exhibit a skill composition gap**, as cascaded inference always outperforms direct inference. The gap ranges from 22.19 points in the case of LLaVA 1.6-Vicuna-13B, to 2.45 points in the case of Qwen2.5-VL-7B. Interestingly, the gap for Qwen2.5-VL-72B is only 0.39 points, which does not indicate a meaningful skill composition gap.

While every model exhibits a cross-modality skill composition gap, both small and large models of the Molmo and Qwen2.5-VL families exhibit significantly smaller gaps than other models. Manual inspection of the *Pixmo Ask Model Anything* (Deitke et al., 2024) dataset used to train Molmo shows examples of math problems presented in images, which provides explicit training data for the skill composition we are testing in this task. For Qwen2.5-VL (Bai et al., 2025), the technical report also mentions the usage of invoices, forms, receipts, table/diagram corpora for training, making the model naturally reason over rendered textual information. This stresses the importance of knowing the training recipes of the models we test, as they allow us to track capabilities to training data, and distinguish “novel” skill composition from skills a model has explicitly been trained on.

Furthermore, the results for the oracle setup are significantly higher than the cascaded ones, even for the most capable models, meaning that the MLLM’s OCR errors account for substantial degradation in final performance, which is further detailed in Appendix C.

### 5.2. Results for Task 2: Object counting

Object counting can be decomposed into two steps: object detection or pointing (a visual skill) and counting the occurrences of a specific string in a text (a textual skill). To implement Task 2, we use the CV-Bench (Tong et al., 2024) benchmark, which, given an image, prompts the model to count a specific type of object. As CV-Bench does not provide the object annotations of the images, we can neither evaluate MLLMs with oracle inference, nor individually analyze the visual and textual skills of the models. To better study these aspects, we introduce a new dataset: COCO-Count. This dataset is built from the 2017 validation split of COCO (Lin et al., 2014), and consists of images paired with annotated bounding boxes and points and the corresponding number of instances of a target object (see Appendix F.2 for details).

Results for each benchmark can be seen in Table 2. We only show the results of models that have been trained for the required visual and textual skills (as discussed in §4.1 only Molmo and Qwen2.5-VL have the object detection skill). Results for the re-

Model	Setup	Datasets					Avg
		GSM8K	ARC-E	ARC-C	MMLU	MATH	
LLaVA 1.6-Vicuna-13B	Oracle	29.11	83.83	70.05	51.73	4.60	47.86
	Direct	8.79	28.82	27.21	26.64	<b>3.20</b>	18.93
	Cascaded	<b>26.91</b>	<b>72.64</b>	<b>60.75</b>	<b>43.68</b>	1.60	<b>41.12</b>
LLaVA 1.6-Mistral-7B	Oracle	42.53	87.42	75.77	53.01	12.80	54.31
	Direct	11.75	51.51	45.05	31.68	5.60	29.12
	Cascaded	<b>36.77</b>	<b>75.71</b>	<b>62.37</b>	<b>46.49</b>	<b>7.00</b>	<b>45.67</b>
Llama 3.2 11B	Oracle	84.23	94.69	87.71	72.08	46.20	76.98
	Direct	63.59	69.23	61.43	46.37	36.40	55.40
	Cascaded	<b>81.05</b>	<b>94.74</b>	<b>87.88</b>	<b>68.21</b>	<b>40.40</b>	<b>74.46</b>
Molmo-7B	Oracle	70.28	90.61	82.08	60.88	27.40	66.25
	Direct	60.27	77.10	66.89	45.07	16.00	53.07
	Cascaded	<b>66.87</b>	<b>86.57</b>	<b>76.02</b>	<b>56.67</b>	<b>23.00</b>	<b>61.83</b>
Molmo-72B	Oracle	93.10	98.53	95.05	81.93	62.40	86.20
	Direct	87.64	93.64	88.14	67.36	40.40	75.44
	Cascaded	<b>92.27</b>	<b>97.81</b>	<b>93.69</b>	<b>75.61</b>	<b>45.00</b>	<b>80.88</b>
Qwen2.5-VL-7B	Oracle	85.14	96.21	88.82	71.32	65.60	81.42
	Direct	79.45	94.78	86.17	66.06	<b>61.80</b>	77.65
	Cascaded	<b>83.85</b>	<b>96.00</b>	<b>88.57</b>	<b>70.28</b>	<b>61.80</b>	<b>80.10</b>
Qwen2.5-VL-72B	Oracle	95.75	99.03	96.67	85.55	80.40	91.48
	Direct	95.22	98.48	95.39	80.94	<b>77.20</b>	89.45
	Cascaded	<b>95.30</b>	<b>98.57</b>	<b>96.59</b>	<b>82.94</b>	75.80	<b>89.84</b>

Table 1: Exact-match accuracy of MLLMs for Task 1 (reasoning over rendered text) under three different inference setups on each benchmark. Oracle is provided as an upper bound reference for the cascaded inference. The best performance for each model under the two main inference setups (cascaded and direct) is shown in **bold**. The difference between the cascaded and direct results reveals the skill composition gap, which is significant for almost all MLLMs.

mained models can be found in Appendix D. We find that Qwen2.5-VL models exhibit a clear skill composition gap, whereas Molmo models do not. To investigate this further, we measured the performance of each model on each individual skill to determine whether the absence of a composition gap is due to effective skill integration, or simply because the model fails to perform one or both skills correctly. We use the COCO-Count dataset for this analysis, as it is the only dataset that includes annotated bounding boxes and points, which are necessary for assessing the models’ counting ability in isolation.

Table 3 summarizes these results. It can be seen that although Molmo has been reported to be trained on both skills, Molmo-7B shows poor performance in each one individually with the accuracy below 50%. Molmo-72B performs better; however, its accuracy remains below 80%, which can still be considered low, especially for the string counting textual skill, where Qwen2.5-VL models achieve nearly perfect performance. Therefore, Molmo models’ low score in the cascaded setup cannot be attributed to a lack of skill composition, but rather to insufficient competence in the individ-

ual skills themselves.

In summary, we again observe a clear skill composition gap for both Qwen2.5-VL models. This gap is not evident for the Molmo models, which perform poorly on the individual skills themselves. Further analysis reveals that both Molmo models struggle with the simple skill of textual counting, likely contributing to their weak performance in the cascaded setting. Interestingly, Molmo models perform comparably well under direct inference, potentially because they were directly trained on this task.

### 5.3. Results for Task 3: Card playing

Task 3 is formulated as the sequential composition of image recognition (visual skill) and numerical reasoning (textual skill). To this end, we design two different card games similar to Chu et al. (2025). The first card game presents a sequence of four randomly ordered poker cards. The task is to determine the correct position of a new card such that the ascending order of the sequence is preserved. The second game also consists of four randomly selected cards, but unordered. The objective is to

Model	Setup	Datasets		Avg
		CV-Bench	COCO-Count	
Molmo-7B	Oracle	N/A	38.11	N/A
	Direct	<b>60.53</b>	<b>56.62</b>	<b>58.58</b>
	Cascaded	33.12	16.07	24.60
Molmo-72B	Oracle	N/A	79.68	N/A
	Direct	64.47	<b>57.53</b>	<b>61.00</b>
	Cascaded	<b>65.61</b>	53.29	59.45
Qwen2.5-VL-7B	Oracle	N/A	99.53	N/A
	Direct	65.48	66.36	65.92
	Cascaded	<b>70.68</b>	<b>70.31</b>	<b>70.50</b>
Qwen2.5-VL-72B	Oracle	N/A	99.91	N/A
	Direct	69.16	65.17	67.17
	Cascaded	<b>70.81</b>	<b>70.44</b>	<b>70.63</b>

Table 2: Exact-match accuracy of MLLMs for Task 2 (object counting) under three different inference setups on each benchmark. Oracle is provided as an upper bound reference for the cascaded inference (when possible). The best performance for each model under the two main inference setups (cascaded and direct) is shown in **bold**. The difference between the cascaded and direct results reveals the skill composition gap, which is significant for both Qwen2.5-VL models, but non-existent for both Molmo models (see text for details).

Model	Skill	COCO-Count
Molmo-7B	Obj. detection (visual)	46.82
	Str. counting (textual)	38.11
	Cascaded	16.07
Molmo-72B	Obj. detection (visual)	65.87
	Str. counting (textual)	79.68
	Cascaded	53.29
Qwen2.5-VL-7B	Obj. detection (visual)	67.65
	Str. counting (textual)	99.53
	Cascaded	70.31
Qwen2.5-VL-72B	Obj. detection (visual)	70.76
	Str. counting (textual)	99.91
	Cascaded	70.44

Table 3: Accuracy of the MLLMs for the visual skill of object detection and the textual skill of string counting, which are involved in Task 2 (object counting). We also provide the cascaded result as reference, as the combination of both skills.

compute the sum of the card values according to specific conditions, where not only the card numbers but also the suits and colors play a crucial role in solving the task (details in Appendix F.3).

Table 4 presents the results for this task, where the *Sort* column reports performance on the first game and the *Sum* column corresponds to the second game. As in previous tasks, we observe a skill composition gap for every model and game (with the exception LLaVA 1.6-Mistral-7B on Sum, where performance is very poor for both inference setups). Notably, this gap is quite pronounced even for mod-

Model	Setup	Datasets		Avg
		Sort	Sum	
LLaVA 1.6-Vicuna-13B	Oracle	22.90	27.70	25.30
	Direct	21.10	2.80	11.95
	Cascaded	<b>21.60</b>	<b>3.80</b>	<b>12.70</b>
LLaVA 1.6-Mistral-7B	Oracle	45.20	38.90	42.05
	Direct	30.70	<b>4.40</b>	17.55
	Cascaded	<b>39.60</b>	3.90	<b>21.75</b>
Llama 3.2 11B	Oracle	59.60	62.10	60.85
	Direct	44.60	11.30	27.95
	Cascaded	<b>57.60</b>	<b>44.20</b>	<b>50.90</b>
Molmo-7B	Oracle	53.60	52.30	52.95
	Direct	47.30	30.30	38.80
	Cascaded	<b>49.80</b>	<b>43.80</b>	<b>46.80</b>
Molmo-72B	Oracle	93.30	75.90	84.60
	Direct	87.90	70.80	79.35
	Cascaded	<b>92.70</b>	<b>77.90</b>	<b>85.30</b>
Qwen2.5-VL-7B	Oracle	75.00	52.60	63.80
	Direct	57.60	51.80	54.70
	Cascaded	<b>74.40</b>	<b>52.80</b>	<b>63.60</b>
Qwen2.5-VL-72B	Oracle	97.30	99.70	98.50
	Direct	96.00	99.40	97.70
	Cascaded	<b>97.20</b>	<b>99.60</b>	<b>98.40</b>

Table 4: Exact-match accuracy of MLLMs for Task 3 under three different inference setups on each benchmark. Oracle serves as an upper bound for cascaded inference. The best results for each model under the direct and cascaded inference setups are shown in **bold**. The difference between the cascaded and direct results reveals the skill composition gap, which is significant for all MLLMs.

els that showed a smaller gap in Task 1, such as Molmo-7B, Molmo-72B and Qwen2.5-VL-7B. We hypothesize that this may be due to the novelty of this task, which makes it less likely that these models were directly trained on it. However, in the case of Qwen2.5-VL-72B the gap is not very big suggesting that this model is better at composing the required skills for this task.

Furthermore, a gap also exists between the oracle and direct setups. For some models, such as Qwen2.5-VL-7B and Qwen2.5-VL-72B, this gap is minimal, indicating that the limiting factor is its textual reasoning capability. We further experiment on the individual skill performance in Appendix E, confirming visual understanding for this task is perfect in the case of both Qwen2.5-VL models.

## 6. Mitigation strategies

Our results so far reveal a consistent lack of optimal skill composition across tasks. This observation naturally leads us to the question of *How can we mitigate this gap?* To address this, we explore two strategies. In §6.1, we introduce an approach

based on Composition-specific Chain-of-Thought (CoT) prompting. In §6.2, we explore a fine-tuning strategy to enhance skill composition.

### 6.1. Composition-specific Chain-of-Thought prompting

As the first mitigation strategy, we opted for Chain-of-Thought (CoT) prompting (Wei et al., 2022). Specifically, we provide the model with an image and a CoT prompt that explicitly states which skills to combine and in what order (see Appendix B for the evaluated prompts). The underlying idea is that this approach should work in much the same way as the cascaded setup, since both aim to sequentially apply different skills, except that in this case the entire reasoning process is guided within a single prompt and a single model call.

Table 5 reports the results for CoT prompting alongside the previous inference setups. The effectiveness of this strategy appears to be model-dependent. For example, Qwen2.5-VL-7B, Qwen2.5-VL-72B, Molmo-7B and Molmo-72B show consistent improvements across most datasets, outperforming the direct setup. Similarly, both LLaVA models mostly outperform the direct setup, but their gain is overall more significant. For instance, there is an improvement of 36.33 points for LLaVA-Vicuna and 23.11 points for LLaVA-Mistral on the ARC-Easy dataset. In contrast, Llama fails to make any meaningful gains, and tends to perform at or below the direct setup.

Although this mitigation strategy performs relatively well, showing improvements over the direct setup in most datasets, it is still far from the performance achieved by the cascaded setup in nearly all cases. Furthermore, the approach is not scalable to arbitrary tasks, as it requires designing prompts manually that explicitly specify which skills the model should apply and in what order. Also, as shown in Appendix G.1 prompt design itself is a challenging process, with inconsistent effectiveness across different models with distinct prompts. This reinforces the idea that prompt engineering is non-trivial and model-sensitive, highlighting the need to explore alternative strategies for achieving effective skill composition.

Moreover, to determine whether the performance improvements observed in Table 5 are due to more effective skill usage with CoT prompting, we manually analyze the generations of two models on two datasets (more details in Appendix G.2). The analysis shows that CoT prompting enables MLLMs to compose skills as requested, whereas direct inference shows a different strategy to solve the same tasks. Thus, the performance improvements we observe in Table 5 can be attributed to a better skill usage with CoT prompting.

### 6.2. Fine-tuning on a cross-modal task

We also explored training strategies for skill composition. More concretely, we have sought to answer the following question: (1) how does fine-tuning on a task that requires cross-modal skill composition help reduce the skill composition gap on the same task? and (2) to what extent does this mitigation strategy generalize to different tasks that also require cross-modal skill composition?

To this end we evaluated the full matrix of fine-tuning in each dataset and evaluating on all others. Specifically, we fine-tuned the Llama 3.2 11B model since it achieves good results and shows a significant gap between direct and cascaded inference, while not being the lowest or highest performing model. For fine-tuning, we followed the approach of Chu et al. (2025) as a reference and fine-tuned on all datasets except MMLU and CV-Bench since they lack training splits (more details in Appendix A.2).

The goal of our fine-tuning is to reduce or remove the cross-modal skill-composition gap, i.e. we want the model with the visual input (the Direct setup) to perform as well as with the textual input (the Oracle upper bound). To this end, we rendered each training dataset into images and fine-tuned the MLLM directly on these visual inputs. However, the textual part of the task requires reasoning capabilities, and the selected datasets do not include explicit textual reasoning annotations. To address this limitation, we proceed as follows: (i) we prompt the text-only version of the model with the textual description of the problem, (ii) we store the generated solutions, and (iii) we pair the corresponding image inputs with these generated solutions to construct a new training dataset.

This approach effectively trains the MLLM to achieve comparable performance when conditioned on visual inputs as when conditioned on textual descriptions, and has the advantage of training on in-distribution model outputs. We chose to fine-tune on the natural text-only model distribution since it preserves the model's existing reasoning behavior. This methodology allows for studying whether the skill composition gap is reduced. In contrast, rejection-based fine-tuning has the potential to improve the model's reasoning ability, which could result in confounding the analysis.

As shown in Table 6, fine-tuning and evaluating on the same dataset improves direct inference performance. However, this improvement is not sufficient to close the skill composition gap for Task 1, since none of the fine-tuned models surpasses the cascaded inference results of the original model. Analyzing Task 2 results, we observe that most of the fine-tunings outperform the cascaded inference, suggesting that the mitigation strategy pays off. However, it is worth noting that Llama was not originally trained for object detection and counting;

Model	Setup	Task 1					Task 2		Task 3		Avg
		GSM8K	ARC-E	ARC-C	MMLU	MATH	CV-B	COCO	Sort	Sum	
LLaVA 1.6-Vicuna-13B	Direct	8.79	28.82	27.21	26.64	3.20			21.10	2.80	16.94
	Cascaded	<b>26.91</b>	<b>72.64</b>	<b>60.75</b>	<b>43.68</b>	1.60	N/A	N/A	<b>21.60</b>	<b>3.80</b>	<b>33.00</b>
	CoT	<u>12.28</u>	<u>65.15</u>	<u>52.04</u>	<u>37.76</u>	<b>6.60</b>			<u>18.40</u>	<b>3.90</b>	<u>28.02</u>
LLaVA 1.6-Mistral-7B	Direct	11.75	51.51	45.05	31.68	5.60			30.70	<b>4.40</b>	25.81
	Cascaded	<b>36.77</b>	<b>75.71</b>	<b>62.37</b>	<b>46.49</b>	<u>7.00</u>	N/A	N/A	<b>39.60</b>	<b>3.90</b>	<b>38.83</b>
	CoT	<u>29.41</u>	<u>74.62</u>	<u>60.84</u>	<u>42.51</u>	<b>7.60</b>			<u>39.40</u>	<b>4.40</b>	<u>36.97</u>
Llama 3.2 11B	Direct	63.59	69.23	61.43	46.37	36.40			44.60	11.30	47.56
	Cascaded	<b>81.05</b>	<b>94.74</b>	<b>87.88</b>	<b>68.21</b>	<b>40.40</b>	N/A	N/A	<b>57.60</b>	<b>44.20</b>	<b>67.73</b>
	CoT	<u>68.54</u>	68.18	59.64	<u>48.23</u>	35.20			<u>47.30</u>	<u>13.10</u>	<u>48.60</u>
Molmo-7B	Direct	60.27	77.10	66.89	45.07	<u>16.00</u>	<b>60.53</b>	<b>56.62</b>	<u>47.30</u>	30.30	<u>51.12</u>
	Cascaded	<b>66.87</b>	<b>86.57</b>	<b>76.02</b>	<b>56.67</b>	<b>23.00</b>	33.12	16.07	<b>49.80</b>	<b>43.80</b>	<b>50.21</b>
	CoT	<u>62.70</u>	<u>79.37</u>	<u>68.08</u>	<u>46.14</u>	14.00	<u>48.22</u>	<u>50.00</u>	27.50	<u>40.10</u>	<u>48.46</u>
Molmo-72B	Direct	87.64	93.64	88.14	67.36	<u>40.40</u>	64.47	<u>57.53</u>	<u>87.90</u>	<u>70.80</u>	73.10
	Cascaded	<b>92.27</b>	<b>97.81</b>	<b>93.69</b>	<b>75.61</b>	<b>45.00</b>	<u>65.61</u>	53.29	<b>92.70</b>	<b>77.90</b>	<b>73.10</b>
	CoT	<u>91.96</u>	<u>95.79</u>	<u>89.33</u>	<u>68.49</u>	<b>45.00</b>	<b>68.40</b>	<b>67.51</b>	85.80	<u>67.20</u>	<u>75.50</u>
Qwen2.5-VL-7B	Direct	<u>79.45</u>	94.78	86.17	66.06	<b>61.80</b>	65.48	<u>66.36</u>	57.60	51.80	69.94
	Cascaded	<b>83.85</b>	<b>96.00</b>	<b>88.57</b>	<b>70.28</b>	<b>61.80</b>	<b>70.68</b>	<b>70.31</b>	<b>74.40</b>	52.80	<b>74.30</b>
	CoT	<u>79.38</u>	<u>95.37</u>	<u>86.68</u>	<u>67.37</u>	<u>58.40</u>	<u>67.51</u>	66.20	<u>67.90</u>	<b>54.10</b>	<u>71.43</u>
Qwen2.5-VL-72B	Direct	<u>95.22</u>	98.48	95.39	80.94	<b>77.20</b>	69.16	65.17	<u>96.00</u>	<u>99.40</u>	86.33
	Cascaded	<b>95.30</b>	<u>98.57</u>	<b>96.59</b>	<u>82.94</u>	<u>75.80</u>	<u>70.81</u>	<b>70.44</b>	<b>97.20</b>	<b>99.60</b>	<b>87.47</b>
	CoT	<u>95.07</u>	<b>98.65</b>	<u>95.65</u>	<b>83.33</b>	<b>77.20</b>	<b>71.57</b>	<u>68.36</u>	95.50	<u>99.40</u>	<u>87.19</u>

Table 5: Results of MLLMs for our three tasks, under the direct and cascaded inferences, as well as the CoT inference (mitigation strategy). Best values per dataset in **bold**, second-best underlined. For all MLLMs, except for Molmo-7B, Cascaded > CoT > Direct, which shows CoT mitigates the skill composition gap, but does not perform as well as cascaded.

Ft data	Setup	Task 1					Task 2		Task 3	
		GSM8K	ARC-E	ARC-C	MMLU	MATH	CV-B	COCO	Sort	Sum
None	Dir.	63.59	69.23	61.43	46.37	36.40	61.55	60.00	44.60	11.30
	Cas.	81.05	94.74	87.88	68.21	40.40	36.67	29.48	57.60	44.20
GSM8K	Dir.	66.87 (+3.28)	71.04 (+1.81)	63.40 (+1.97)	47.48 (+1.11)	32.80 (-3.60)	<b>64.85</b> (+3.30)	<b>60.26</b> (+0.26)	46.70 (+2.00)	11.60 (+0.30)
ARC-E/C	Dir.	66.94 (+3.35)	79.88 (+10.65)	65.10 (+3.67)	50.08 (+3.71)	34.80 (-1.60)	<b>64.72</b> (+3.17)	<b>60.90</b> (+0.90)	45.70 (+1.00)	9.70 (-1.60)
MATH	Dir.	64.97 (+1.38)	73.48 (+4.25)	60.58 (-0.85)	46.27 (-0.10)	38.80 (+2.40)	<b>64.21</b> (+2.66)	<b>59.45</b> (-0.55)	47.20 (+2.50)	8.90 (-2.40)
COCO	Dir.	57.84 (-5.75)	64.94 (-4.29)	58.19 (-3.24)	43.73 (-2.64)	33.00 (-3.40)	<b>46.44</b> (-15.11)	<b>69.09</b> (+9.09)	49.70 (+5.00)	1.30 (-10.00)
Sort	Dir.	68.23 (+4.64)	69.53 (+0.30)	61.60 (+0.17)	46.47 (+0.10)	38.00 (+1.60)	<b>64.34</b> (+2.79)	<b>60.27</b> (+0.27)	<b>63.30</b> (+18.60)	14.40 (+3.10)
Sum	Dir.	66.18 (+2.59)	71.51 (+2.28)	62.20 (+0.77)	47.12 (+0.75)	34.60 (-1.80)	<b>65.74</b> (+4.19)	<b>60.98</b> (+0.98)	<b>60.60</b> (+15.90)	<b>71.90</b> (+60.60)

Table 6: Results of the Llama3.2 11B model for all tasks under the direct setup when fine-tuning in each dataset. Results of the original model under direct (Dir.) and cascaded (Cas.) inferences are also presented as reference (block tagged as "None"). Numbers in parentheses indicate the difference with respect to the direct baseline. Results that surpass the cascaded baseline are in bold.

therefore, this comparison is not entirely fair (the original model’s performance is extremely low). For Task 3, some fine-tunings improve the performance of the model, but only in-distribution training outperforms the cascaded results.

Our results show that skill-composition-specific fine-tuning has notable generalization capabilities beyond the training dataset. We observe improved performance on other datasets within the same task, indicating a successful transfer of task-specific skill composition. Additionally, performance gains extend to datasets from entirely different tasks, suggesting that fine-tuning enhances the model’s broader skill composition capabilities.

Overall, while fine-tuning consistently improves

performance across diverse evaluation scenarios, the skill composition gap is only fully closed when fine-tuning and evaluation are performed on the same dataset.

## 7. Conclusions

In this work, we have studied whether MLLMs are able to optimally compose the skills they acquire during pretraining to solve new tasks. To this end, we introduce three multimodal tasks that can be solved by sequentially composing one visual and one textual skill, in a way that would be trivial to find for a human. We compare the model’s natural performance with a cascaded approach where we

manually force the skill composition, and measure the resulting **cross-modality skill composition gap**.

We find that, somewhat strikingly, a significant skill composition gap exists across almost all measured models and tasks, meaning that MLLMs are not able to optimally compose their existing multi-modal abilities to solve a task, even in this scenario where the required skill composition is almost trivial. Having shown the existence of this skill composition gap, we explore two possible mitigation strategies: a special CoT that explicitly encourages the required skill composition, and a fine-tuning training strategy that enhances skill-composition, but none of them fully bridge the gap with cascaded inference and present different disadvantages.

In conclusion, this work prompts further research into why MLLMs have trouble successfully composing skills across modalities even in simple scenarios, as well as into new training approaches for more robust skill composition and generalization.

## Limitations

Our work currently focuses on the combination of only two skills. However, it would be interesting to incorporate a wider range of skills, both visual and textual, into the tasks. This would be more similar to a real-world scenario in which several skills are required to solve complex problems. Additionally, our work is limited to the evaluation of open-source models. Thus, evaluating commercial models could also be interesting. However, our current evaluation is limited to open-source models because their training data and design choices are reported, making it possible to infer which skills they have likely acquired. This is crucial for accurately assessing skill composition across modalities. While commercial models such as GPT-4 (OpenAI et al., 2024b), Gemini (Team et al., 2023), Reka (Team et al., 2024) or Claude<sup>4</sup> could offer further insights, interpreting the results meaningfully is challenging due to the limited information available about their training data and learned skills.

## Ethics Statement

To the best of our knowledge, this paper does not raise any ethical concerns.

## Acknowledgments

This work has been partially supported by the Basque Government (Research group funding

---

<sup>4</sup><https://www.anthropic.com/claude/sonnet>

IT1570-22) and the Spanish Ministry of Science, Innovation and Universities (DeepKnowledge project PID2021-127777OBC2, Molvi project PID2024-157855OB-C32, HumanAlze project AIA2025-163322-C61 and Geo-R2-LLM project PCI2025-163286 by MICIU/AEI/10.13039/501100011033 and co-financed by the European Union). Paula Ontalvilla holds a PhD grant from the Basque Government (PRE\_2024\_1\_0141).

## 8. Bibliographical References

2023. [Gpt-4v\(ision\) system card](#).

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. [Flamingo: a visual language model for few-shot learning](#).

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

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. 2025. [Sft memorizes, rl generalizes: A comparative study of foundation model post-training](#).

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Wenliang Dai, Nayeon Lee, Boxin Wang, Zhuolin Yang, Zihan Liu, Jon Barker, Tuomas Rintamaki, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. [Nvlm: Open frontier-class multi-modal llms](#).

Yunkai Dang, Kaichen Huang, Jiahao Huo, Yibo Yan, Sirui Huang, Dongrui Liu, Mengxi Gao, Jie

- Zhang, Chen Qian, Kun Wang, Yong Liu, Jing Shao, Hui Xiong, and Xuming Hu. 2024. [Explainable and interpretable multimodal large language models: A comprehensive survey](#). *ArXiv*, abs/2412.02104.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#).
- Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohamadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. 2024. [Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models](#). *CoRR*.
- Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. 2023. [Palm-e: An embodied multimodal language model](#).
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. [A framework for few-shot language model evaluation](#).
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. [The llama 3 herd of models](#). *arXiv preprint arXiv:2407.21783*.
- Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. 2024. [Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks](#). *Advances in Neural Information Processing Systems*, 37:13244–13273.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. [Measuring massive multi-task language understanding](#). *arXiv preprint arXiv:2009.03300*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2024. [Measuring mathematical problem solving with the math dataset, 2021](#). URL <https://arxiv.org/abs/2103.03874>.
- Brenden M Lake and Marco Baroni. 2023. [Human-like systematic generalization through a meta-learning neural network](#). *Nature*, 623(7985):115–121.

- Xiujun Li, Yujie Lu, William Yang Wang, and Yejin Choi. 2024. [Pixelated instructions: Can multi-modal large language models follow printed instructions in images?](#) In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Ji Lin, Hongxu Yin, Wei Ping, Yao Lu, Pavlo Molchanov, Andrew Tao, Huizi Mao, Jan Kautz, Mohammad Shoeybi, and Song Han. 2023. [Vila: On pre-training for visual language models](#). *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26679–26689.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer vision—ECCV 2014: 13th European conference, zurich, Switzerland, September 6–12, 2014, proceedings, part v 13*, pages 740–755. Springer.
- Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.
- Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024b. [Llava-next: Improved reasoning, ocr, and world knowledge](#).
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. [Visual instruction tuning](#).
- Tenglong Liu, Jianxiong Li, Yinan Zheng, Haoyi Niu, Yixing Lan, Xin Xu, and Xianyuan Zhan. 2025. Skill expansion and composition in parameter space. *arXiv preprint arXiv:2502.05932*.
- Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. [Chameleon: Plug-and-play compositional reasoning with large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 43447–43478. Curran Associates, Inc.
- OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O’Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gulemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trębacz, Manas Joglekar, Mark Chen, Marko Tinctor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ash-

bourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024a. [OpenAI o1 system card](#).

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altmenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madeline Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang,

Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lillian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong

- Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024b. [Gpt-4 technical report](#).
- Ahmed H Qureshi, Jacob J Johnson, Yuzhe Qin, Taylor Henderson, Byron Boots, and Michael C Yip. Composing task-agnostic policies with deep reinforcement learning. In *International Conference on Learning Representations*.
- Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, Ruochen Xu, and Tiancheng Zhao. 2025. [Vlm-r1: A stable and generalizable r1-style large vision-language model](#).
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.
- Reka Team, Aitor Ormazabal, Che Zheng, Cyprien de Masson d'Autume, Dani Yogatama, Deyu Fu, Donovan Ong, Eric Chen, Eugenie Lamprecht, Hai Pham, et al. 2024. Reka core, flash, and edge: A series of powerful multimodal language models. *arXiv preprint arXiv:2404.12387*.
- Peter Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Adithya Jairam Vedagiri IYER, Sai Charitha Akula, Shusheng Yang, Jihan Yang, Manoj Middepogu, Ziteng Wang, et al. 2024. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356.
- Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhui Chen. 2025a. [VI-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning](#).
- Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haiyan Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhui Wang, and Gen Luo. 2025b. [InternV3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency](#).
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. 2023. Skill-mix: A flexible and expandable family of evaluations for ai models. In *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*.
- Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, et al. 2024. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. *CoRR*.
- Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. 2024. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pages 169–186. Springer.
- Haoyu Zhao, Simran Kaur, Dingli Yu, Anirudh Goyal, and Sanjeev Arora. 2024. Can models learn skill composition from examples? *Advances in Neural Information Processing Systems*, 37:102393–102427.
- Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. 2025. [Skillweaver: Web agents can self-improve by discovering and honing skills](#).
- Yanli Zhou, Brenden M Lake, and Adina Williams. 2024. Compositional learning of functions in humans and machines. *arXiv preprint arXiv:2403.12201*.

## A. Implementation details

### A.1. Evaluation

We evaluated both multiple-choice and open-ended generation datasets using the transformers library of Hugging Face<sup>5</sup>. For the multiple-choice question datasets (ARC-E, ARC-C, MMLU, and CV-Bench), we explicitly prompted the model to select one of the possible answer choices, as shown in Appendix B. To extract the model’s selected answer, we applied various regular expressions to identify the predicted letter. Finally, before computing exact-match accuracy, both predicted and ground-truth answers were normalized by converting to lowercase and removing leading or trailing whitespace.

For the open-ended generation datasets, we employed different extraction strategies. In the case of GSM8K, we followed the approach proposed by (Gao et al., 2024), using a combination of filters and regular expressions to extract the final answer. For the remaining datasets (MATH, COCO-Count, Sort, and Sum) we also relied on regular expressions, with dataset-specific filtering functions designed to best extract the model’s generated answer in each case. More details and the specific regex expressions can be found in the project’s repository<sup>6</sup>. As with the multiple-choice datasets, exact-match accuracy was computed after normalizing both the predicted and ground-truth answers.

However, for the MATH dataset, we adopted an LLM-based evaluation, in line with a widely-used library<sup>7</sup>. Since both predictions and gold answers often contain  $\LaTeX$  expressions and certain expressions can be written in different but mathematically equivalent ways (e.g.,  $\frac{2}{3}$  and 2/3), string matching is not always a reliable metric. To address this, we used the Reka-Flash (Team et al., 2024) model for evaluation. This choice was supported by an empirical study in which we measured the agreement between Reka-Flash and human-annotated labels on 200 instances, achieving a 94.5% match.

### A.2. Hardware information

Each experiment was not very compute-intensive. All experiments were conducted on an internal cluster equipped with NVIDIA A100-SXM4-80GB GPU. All the experiments were carried out in one A100-SXM4-80GB GPU. However, due to the requirement of a batch size of one, each experiment took between 10 hours and 2 days to complete, depending on the setup. The cascaded setup was the

<sup>5</sup><https://huggingface.co/>

<sup>6</sup><https://github.com/paulaonta/skillComposition>

<sup>7</sup><https://github.com/openai/simple-evals>

most time-consuming, as it involves a two-step approach.

For the fine-tuning we used 4 NVIDIA A100-SXM4-80GB GPUs with a batch size of 2, gradient accumulation of 16, a learning rate of 1e-6, no LoRA, and a dropout rate of 0.1.

## B. Prompts

The prompts used in the experiments are available in the project’s repository<sup>8</sup>. For the design of the prompts we used as a reference different works (Deitke et al., 2024; Gao et al., 2024; Chu et al., 2025) and libraries<sup>9</sup>.

## C. Analysis for Task 1

We report in Table 7 models’ OCR accuracy using Word Error Rate (WER) and Character Error Rate (CER) metrics. WER measures the proportion of **word** level errors (insertions, deletions, or substitutions) relative to the total number of words in the ground truth, while CER performs the same calculation at the **character** level.

These metrics provide complementary insights: CER is more sensitive to minor spelling or formatting errors, whereas WER penalizes any deviation that results in an incorrect word. In practice, WER is often higher than CER, since a single character mistake can cause an entire word to be marked as incorrect.

The lower the value, the better the performance. However, the output is not always a number between 0 and 1. In particular when there are a high number of insertions (for example, when comparing “hello world” and “hello”) the value can be higher than 1.

It can be seen in Table 7 that the newest models, Qwen2.5-VL-7B, Qwen2.5-VL-72B, Molmo-7B and Molmo-72B achieve the lowest WER and CER scores. Llama 3.2 11B also performs well across most datasets, with the exception of MATH, where its error rates are higher than 1. In contrast, the LLaVA models exhibit the highest WER and CER values, particularly in the MMLU and MATH datasets, indicating weaker visual text recognition in these cases.

## D. Analysis for Task 2

Table 8 presents the exact-match accuracy for models that were not trained specifically for object detection, which is the case of LLaVA 1.6-Vicuna-13B,

<sup>8</sup><https://github.com/paulaonta/skillComposition>

<sup>9</sup><https://github.com/openai/simple-evals>

Model	Metric	Datasets				
		GSM8K	ARC-E	ARC-C	MMLU	MATH
LLaVA 1.6-Mistral-7B	WER	0.0753	0.1331	0.2214	0.7993	1.3482
	CER	0.0647	0.1173	0.2017	0.6893	1.0056
LLaVA 1.6-Vicuna-13B	WER	0.0331	0.1000	0.1161	0.4257	1.1251
	CER	0.0268	0.0915	0.1057	0.3537	0.9140
Llama 3.2 11B	WER	0.0610	0.0103	0.0086	0.0741	5.7179
	CER	0.0561	0.0095	0.0076	0.0617	4.6294
Molmo-7B	WER	0.1392	0.1912	0.2175	0.3412	0.4598
	CER	0.1079	0.1584	0.1830	0.2915	0.4280
Molmo-72B	WER	0.0217	0.0555	0.0629	0.2676	0.4119
	CER	0.0112	0.0380	0.0484	0.2193	0.3178
Qwen2.5-VL-7B	WER	0.0071	0.0112	0.0102	0.0864	0.3701
	CER	0.0055	0.0102	0.0098	0.0758	0.2860
Qwen2.5-VL-72B	WER	0.0059	0.0319	0.0401	0.2500	0.2937
	CER	0.0049	0.0322	0.0410	0.2417	0.2434

Table 7: OCR performance of MLLMs across benchmarks using Word Error Rate (WER) and Character Error Rate (CER). Lower scores indicate better OCR accuracy.

LLaVA 1.6-Mistral-7B and Llama 3.2 11B, in Task 2. As anticipated, the proposed skill composition does not work well for this task since the models do not perform correctly in the visual object detection skill as can be shown in Table 9.

### E. Analysis for Task 3

Table 10 presents the image recognition accuracy of the evaluated models in the task of recognizing cards in the Sort dataset.

The **Both** column indicates the percentage of cases in which the model correctly predicted both the card number and suit at the correct index in the sorted sequence. The **Number** and **Suit** columns report accuracy when only the card number or suit, respectively, matches the ground truth at the correct index. The final column, **Non-Critical Card Error**, corresponds to cases where the model fails to return the correct cards, but the output does not negatively affect the final result (e.g., if the ground truth is [1, 4, 6, 7] and the new card is 4 of Hearts, but the model returns [1, 3, 5, 8], the insertion position is still correct even though the specific cards are wrong).

We observe that the most recent models, such as Qwen2.5-VL-7B, Qwen2.5-VL-72B, Molmo-7B and Molmo-72B, achieve near-perfect or perfect performance. Llama 3.2 11B also performs very well, but there is a noticeable gap between its accuracy in Both and its performance on cases where only the position is correct but not the specific card details (Non-Critical Card Error). In contrast, with both LLaVA models the gap is especially pronounced.

This suggests that these models struggle with fully accurate card recognition at the correct positions.

On the other hand, Table 11 shows the accuracy of the models on the Sum dataset when correctly predicting only the cards' number, suit, color, or all attributes. Since the cards are unordered, the index of the correctly predicted card does not matter. It is important to note that the correct prediction of all attributes is crucial for solving the task correctly, as all attributes are needed to compute the total card values.

We see a similar pattern, i.e., Qwen2.5-VL-7B, Qwen2.5-VL-72B, Molmo-7B, Molmo-72B and Llama 3.2 11B perform well and both LLaVA models show a very poor card recognition performance.

Model	Setup	Datasets	
		CV-Bench	COCO-Count
LLaVA 1.6-Vicuna-13B	Oracle	N/A	90.89
	Direct	59.77	52.97
	Cascaded	64.47	63.22
LLaVA 1.6-Mistral-7B	Oracle	N/A	92.18
	Direct	60.91	54.68
	Cascaded	59.89	64.43
Llama 3.2 11B	Oracle	N/A	91.82
	Direct	61.55	60.00
	Cascaded	36.67	29.48

Table 8: Exact-match accuracy of MLLMs that were not trained specifically for object detection in Task 2 under three different setups on each benchmark.

Model	Skill	COCO-Count
LLaVA 1.6-Vicuna-13B	Obj. detection (visual)	65.18
	Str. counting (textual)	90.89
	Cascaded	63.22
LLaVA 1.6-Mistral-7B	Obj. detection (visual)	63.67
	Str. counting (textual)	92.18
	Cascaded	64.43
Llama 3.2 11B	Obj. detection (visual)	26.74
	Str. counting (textual)	91.82
	Cascaded	29.48

Table 9: Accuracy of the MLLMs that were not trained specifically for object detection for the visual skill of object detection and the textual skill of string counting, which are involved in Task 2 (object counting). We also provide the cascaded result as reference, as the combination of both skills.

Model	Match at correct index			Non-Critical Card Error
	Both	Number	Suit	
LLaVA 1.6-Mistral-7B	8.0	54.8	1.0	69.9
LLaVA 1.6-Vicuna-13B	2.0	27.3	7.0	78.0
Llama 3.2 11B	82.2	87.8	89.1	93.9
Molmo-7B	82.4	93.1	82.4	93.1
Molmo-72B	99.9	99.9	100.0	100.0
Qwen2.5-VL-7B	100.0	100.0	100.0	100.0
Qwen2.5-VL-72B	100.0	100.0	100.0	100.0

Table 10: Accuracy of the models in recognizing cards in the Sort dataset.

Model	Match at any index			
	All	Number	Suit	Color
LLaVA 1.6-Mistral-7B	0.0	43.3	8.9	39.4
LLaVA 1.6-Vicuna-13B	1.0	17.9	9.2	39.4
Llama 3.2 11B	73.2	98.3	80.7	78.2
Molmo-7B	77.0	97.9	89.7	89.3
Molmo-72B	99.8	100.0	99.8	99.8
Qwen2.5-VL-7B	100.0	100.0	100.0	100.0
Qwen2.5-VL-72B	99.9	100.0	99.9	99.9

Table 11: Accuracy of the models in recognizing cards in the Sum dataset.

## F. Details about the datasets

### F.1. Source datasets

We obtained all source datasets from the original sources published by the authors. To the best of our knowledge, all the data sources we use are available for non-commercial use.

- **GSM8K** (Cobbe et al., 2021): We obtained the dataset from the official HuggingFace page<sup>10</sup> under a MIT License<sup>11</sup>.
- **ARC Easy and Challenge** (Clark et al., 2018): We obtained the data from the official HuggingFace page<sup>12</sup> where they specify that it is under a Creative Commons Attribution Share Alike 4.0 license.
- **MMLU** (Hendrycks et al., 2020): We got the data from the official GitHub project<sup>13</sup> under a MIT License<sup>14</sup>.
- **MATH** (Hendrycks et al., 2024; Lightman et al., 2023): We obtained the dataset from HuggingFace<sup>15</sup> under a MIT License<sup>16</sup>.
- **CV-Bench** (Tong et al., 2024): We obtained the data from the official HuggingFace page<sup>17</sup> under Apache-2.0 license<sup>18</sup>.

The rendered textual datasets (GSM8K, ARC-E, ARC-C, MMLU and MATH) can be found in the project’s repository.

### F.2. COCO-Count

We used the 2017 validation split of COCO (Lin et al., 2014) from the official project website<sup>19</sup> under a Creative Commons Attribution 4.0 License<sup>20</sup> for several reasons: (1) it is an open-source dataset with annotated bounding boxes for each object, and (2) it does not overlap with the other dataset we used, CV-Bench.

<sup>10</sup><https://huggingface.co/datasets/openai/gsm8k>

<sup>11</sup><https://github.com/openai/grade-school-math?tab=License-1-ov-file>

<sup>12</sup>[https://huggingface.co/datasets/allenai/ai2\\_arc](https://huggingface.co/datasets/allenai/ai2_arc)

<sup>13</sup><https://github.com/hendrycks/test?tab=readme-ov-file>

<sup>14</sup><https://github.com/hendrycks/test?tab=MIT-1-ov-file>

<sup>15</sup><https://huggingface.co/datasets/HuggingFaceH4/MATH-500>

<sup>16</sup><https://github.com/openai/prm800k/tree/main?tab=MIT-1-ov-file>

<sup>17</sup><https://huggingface.co/datasets/nyu-visionx/CV-Bench>

<sup>18</sup><https://github.com/cambrian-mlm/cambrian?tab=Apache-2.0-1-ov-file>

<sup>19</sup><https://cocodataset.org/#download>

<sup>20</sup><https://cocodataset.org/#termsofuse>

Based on this, we created a dataset in which each image is annotated with a single object instance in the format required by each model. The dataset contains a total of 4,952 instances. Since each image may contain multiple objects, we randomly selected only one object per instance to avoid creating an overly large dataset, while still preserving the overall distribution of object counts per image. In other words, we maintained the distribution of the number of objects to be counted, ensuring that models are not always predicting categories with only a single occurrence.

In the case of **Molmo**, instead of using bounding boxes, the model relies on a custom HTML-like format to point to objects. The `points` field refers to the center point of each object, which we computed from the bounding box coordinates. For example, for instance ID 397133, the object “people” is represented as follows. Since there are two `x` and two `y` values, it indicates that there are two people in the image:

```
"397133": {
  "category": "people",
  "points": "<point x1=443.365"
y1=208.73000000000002"
x2=31.08" y2=281.195"
alt="people">people</point>",
  "count": 2,
  "filename": "000000397133.jpg"
}
```

By contrast, for the other models, we used a JSON format consistent with the structure of their outputs. For the same instance, the annotation would look like this, where each object is associated with a separate bounding box:

```
"397133": {
  "count": 2,
  "category": "people",
  "points": [
    {
      "label": "people",
      "bounding_box": [
        388.66,
        69.92,
        498.07000000000005,
        347.54
      ]
    },
    {
      "label": "people",
      "bounding_box": [
        0.0,
        262.81,
        62.16,
        299.58
      ]
    }
  ],
  "filename": "000000397133.jpg"
}
```

Therefore, in the case of Molmo, the model's counting ability can be evaluated based on its ability to infer the number of distinct coordinate pairs (i.e., different  $x$  or  $y$  values). For the other models, performance is measured by their ability to correctly detect and count the number of distinct bounding boxes. More details and the dataset can be found in the project's repository.

### F.3. Card game

To create this dataset, we used the work of (Chuet al., 2025) as a reference which has a MIT license. We constructed two datasets, both based on poker cards sourced from the aforementioned work. In all cases, we excluded face cards (Queen, Jack, and King) since they can be interpreted in different ways (e.g., all as 10, or as 11, 12, and 13). This exclusion simplifies the task and ensures consistent interpretation.

The datasets are designed to test different reasoning skills: the **Sort** dataset focuses on ordinal reasoning, while the **Sum** dataset tests arithmetic reasoning based on visual card attributes. The dataset can be found in the project's repository.

**Sort Dataset.** We randomly selected four cards and arranged them in ascending order based on their numerical values. While duplicate numbers are allowed, each card (i.e., unique number-suit pair) appears at most once. When two cards share the same number, their relative order is arbitrary, as suit is not considered in the sorting criteria.

After forming the initial set of four cards, we randomly selected a fifth card to query. This card was chosen such that it did not appear in the original image and its number did not match any of the cards in the image, to avoid ambiguity. Using this procedure, we created a dataset of 1,000 instances, each consisting of an image showing four sorted cards, a queried card, and the correct position where that card should be inserted. Figure 2 shows an example.

**Sum Dataset.** For this dataset, we also randomly selected four cards, but the order is not relevant. As with the Sort dataset, each card is unique in terms of number-suit combination, although duplicate numbers are allowed.

The task is to compute the sum of the four cards using a set of predefined rules, which test the model's ability to recognize different visual features and to and reason about it:

- Number cards (2–10) have a base value equal to their face value.
- Ace = 1.
- Spades: add 5 bonus points to their base value.

- Clubs with even numbers: subtract 2 from their base value.
- Red cards (Hearts and Diamonds): their value is twice the base value.

We generated 1,000 instances, each consisting of an image with four unordered cards and the correct sum based on the rules above. Figure 3 shows an example.

## G. Composition-specific CoT strategy

### G.1. Scalability

To design the CoT mitigation strategy prompts, we employed prompt engineering techniques. While we observed consistent results across all models and datasets, the Sort dataset is an exception. As shown in Table 12, we experimented with several CoT prompt variations (shown in Table 13) for the Sort dataset; however, they did not produce consistent performance gains across all models.

### G.2. Output analysis

We analyzed some of the model outputs to understand how and why the CoT strategy improves results. This analysis was conducted for the LLaVA 1.6-Mistral-7B model on GSM8K dataset, as it showed one of the largest gains from CoT prompting compared to direct answers. In contrast, we also examined Qwen2.5-VL-7B on CV-Bench dataset, which exhibited the smallest improvement.

In the case of **LLaVA 1.6-Mistral-7B** we analyzed 50 random samples where the CoT strategy produced correct results, while the direct inference failed. We concluded that CoT prompting does change the behavior of the MLLM, which composes the two skills as instructed. In direct inference, the MLLM uses another strategy, trying to solve the reasoning problem directly over the image. Due to this strategy, we observe that: i) the MLLM commits mistakes when gathering the visual information, and ii) the reasoning step is usually simpler than with the CoT version (see the example 4). It is worth mentioning that both behaviors are repeated consistently across all the analyzed instances.

For **Qwen2.5-VL-7B**, we analyzed 66 randomly selected samples where the CoT strategy produced correct results, while the direct inference failed. In this case, we observe again a clear difference between the behavior of the MLLM in direct and CoT setups as it can be seen in 5. While direct inference tries to analyze the whole image region-by-region, providing short and descriptive captions per image region, CoT prompting makes the model focus only on the target objects and leverages the object detection skill. This again shows that the model is more capable than initially evident.

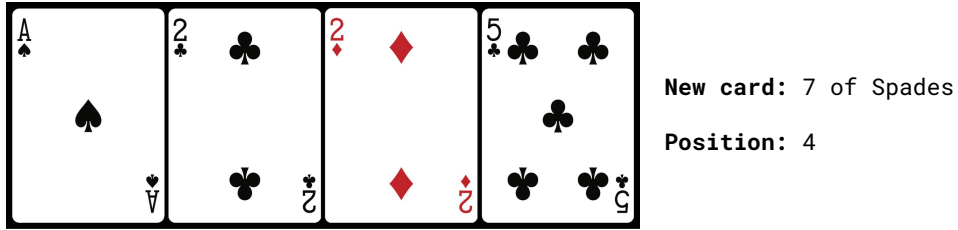


Figure 2: Example from the **Sort** dataset.

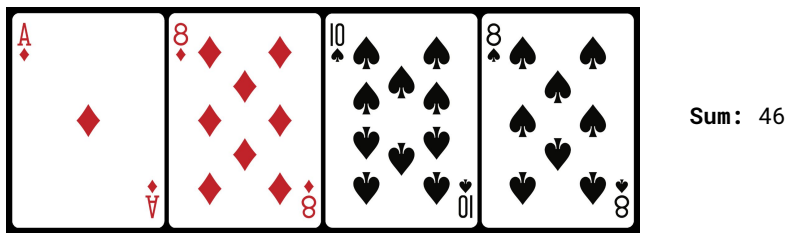


Figure 3: Example from the **Sum** dataset.

Model	Direct	CoT					Cascaded
		P1	P2	P3	P4	P5	
LLaVA 1.6-Mistral-7B	30.70	<b>38.20</b>	<b>33.40</b>	<b>34.30</b>	<b>38.50</b>	<b>35.30</b>	38.40
LLaVA 1.6-Vicuna-13B	19.70	18.70	<b>21.30</b>	16.90	10.10	13.40	20.90
Llama 3.2 11B	44.70	<b>48.30</b>	<b>47.70</b>	43.40	42.20	42.00	56.90
Molmo-7B	42.40	26.50	31.40	39.40	<b>51.30</b>	<b>42.70</b>	49.40
Qwen2.5-VL-7B	58.40	<b>67.50</b>	56.20	54.60	54.00	55.90	75.50

Table 12: Results of the MLLMs with different CoT prompts. In **bold** appears the results in which there is an improvement respect the direct setup results.

Dataset	Prompt
P1	<p>You are observing 4 playing cards sorted in ascending order by their numeric value. Note that A counts as '1'. First, identify the cards and their values. Then, sort them in ascending order.</p> <p>Now, consider a new card: <code>&lt;cards&gt;</code></p> <p>Where should this card be inserted in the sequence so that the final list remains in ascending order?</p> <p>Please answer with the index position (0 to 4), where 0 is before the first card, 1 is between the first and second, and so on, up to 4 which means after the last card.</p> <p>The last line of your response should be of the form <code>Answer: \$ANSWER</code> (without quotes) where <code>\$ANSWER</code> is the position of the card.</p>
P2	<p>You are observing 4 playing cards sorted in ascending order by their numeric value. Note that A counts as '1'. First, identify the cards and their values.</p> <p>Now, consider a new card: <code>&lt;cards&gt;</code></p> <p>Where should this card be inserted in the sequence so that the final list remains in ascending order?</p> <p>Please answer with the index position (0 to 4), where 0 is before the first card, 1 is between the first and second, and so on, up to 4 which means after the last card.</p> <p>The last line of your response should be of the form <code>Answer: \$ANSWER</code> (without quotes) where <code>\$ANSWER</code> is the position of the card.</p>
P3	<p>You are looking at an image showing 4 playing cards sorted in ascending order by their numeric value. First, identify the cards and their values. Note that A (Ace) counts as '1', number cards (2–10) count as their face value.</p>

Continued on next page

Table 13 – continued from previous page

Dataset	Prompt
	<p>Now, consider a new card: <code>&lt;cards&gt;</code></p> <p>Where should this card be inserted in the sequence so that the final list remains in ascending order?</p> <p>Please answer with the index position (0 to 4), where 0 is before the first card, 1 is between the first and second, and so on, up to 4 which means after the last card.</p> <p>The last line of your response should be of the form <code>Answer: \$ANSWER</code> (without quotes) where <code>\$ANSWER</code> is the position of the card.</p>
<b>P4</b>	<p>You are looking at an image showing 4 playing cards sorted in ascending order by their numeric value. Note that A (Ace) counts as '1'.</p> <p>First, identify the cards and their values in the order they appear in the image.</p> <p>Now, consider a new card: <code>&lt;cards&gt;</code></p> <p>Where should this card be inserted in the sequence so that the final list remains in ascending order?</p> <p>Please answer with the index position (0 to 4), where 0 is before the first card, 1 is between the first and second, and so on, up to 4 which means after the last card.</p> <p>The last line of your response should be of the form <code>Answer: \$ANSWER</code> (without quotes) where <code>\$ANSWER</code> is the position of the card.</p>
<b>P5</b>	<p>You are observing 4 playing cards sorted in ascending order by their numeric value. Note that A counts as '1'.</p> <p>First, identify the cards and their values in the order they appear in the image. For each card identify:</p> <ol style="list-style-type: none"> <li>1. The rank: A (Ace), 2–10</li> <li>2. The suit: Diamonds, Hearts, Spades, Clubs</li> </ol> <p>Now, consider a new card: <code>&lt;cards&gt;</code></p> <p>Where should this card be inserted in the sequence so that the final list remains in ascending order?</p> <p>Please answer with the index position (0 to 4), where 0 is before the first card, 1 is between the first and second, and so on, up to 4 which means after the last card.</p> <p>The last line of your response should be of the form <code>Answer: \$ANSWER</code> (without quotes) where <code>\$ANSWER</code> is the position of the card.</p>

Table 13: Prompts used in the CoT mitigation strategy for Task 3.

### Example of LLaVA 1.6-Mistral-7B

**Example:** Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

**Direct inference answer:**

The image presents a mathematical problem involving a market where a farmer sells eggs and muffins. The farmer has three types of eggs: fresh, medium, and large. The price of fresh eggs is \$2 per dozen, while medium and large eggs are sold at the same price per dozen. The farmer also sells muffins for \$1.50 each.

To solve this problem, we need to find the total cost of buying a dozen of each type of egg and one muffin.

First, let's calculate the cost of a dozen of each type of egg:

- Fresh eggs: \$2 per dozen
- Medium and large eggs: \$1.50 per dozen

Now, let's calculate the total cost of buying a dozen of each type of egg:

- Fresh eggs: \$2
- Medium and large eggs: \$3 (two types at \$1.50 each)

Add the cost of the muffin:

- Total: \$2 + \$3 + \$1.50 = \$6.50

The final answer is **\$6.50**.

**CoT strategy answer:**

The text content of the math problem image is:

"Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?"

To solve this problem, we need to find out how many eggs Janet has left after eating and baking.

Janet eats 3 eggs and uses 4 eggs for baking:

$$3 + 4 = 7 \text{ eggs per day}$$

Subtract from the total laid:

$$16 - 7 = 9 \text{ eggs left to sell}$$

Multiply by the price per egg:

$$9 * \$2 = \$18$$

The final answer is **\$18**.

**True answer:** 18

Figure 4: Example generation from LLaVA 1.6-Mistral-7B using the direct inference setup and the CoT strategy on the GSM8K dataset.

**Example:**



How many bottles are in the image? Select from the following choices.

- (A) 0
- (B) 1
- (C) 3
- (D) 2

**Direct inference answer:**

To determine the number of bottles in the image, let's analyze the objects present:

1. There is a table with a decorative item on it that appears to be a decanter or carafe with a stopper, which typically holds liquid.
2. There are no other objects that resemble bottles.

Based on this analysis, there is only one object that could be considered a bottle.

Answer: **D**

**CoT strategy answer:**

To answer this question, I'll follow the two-step approach:

1. Detect all bottles in the image:
  - There is a clear glass bottle with a stopper on a small table in front of the chairs.
2. Count the detected bottles:
  - Only one bottle is visible in the image.

Based on the count, the correct answer is:

Answer: **B**

**True answer:** B

Figure 5: Example generation from Qwen2.5-VL-7B using the direct inference setup and the CoT strategy on the CV-Bench dataset.