V-GIFT: Boosting Visual Instruction Tuning with Self-Supervised Guidance

Sophia Sirko-Galouchenko[1,2] **Monika Wysoczańska**[1] **Andrei Bursuc**[1]
Nicolas Thome[4,3,2] **Spyros Gidaris**[1]

[1]Valeo.ai [2]Sorbonne Université, CNRS, ISIR, F-75005 Paris, France
[3]Institut universitaire de France (IUF) [4] ILLS, CNRS, Montreal, QC H2S 3H1

Abstract

Multimodal large language models (MLLMs) perform well on many vision-language tasks but often struggle with vision-centric problems that require fine-grained visual perception. Recent evidence suggests that this limitation arises not from weak visual representations, but from under-utilization of visual information during instruction tuning, where many tasks can be partially solved using language priors alone. We propose a simple and lightweight approach that augments visual instruction tuning with a small number of visually grounded self-supervised tasks expressed as natural language instructions. By reformulating classical self-supervised pretext tasks, such as rotation prediction, color matching, and cross-view correspondence, as image–instruction–response triplets, we introduce supervision that cannot be solved without relying on visual evidence. Our approach requires no human annotations, no architectural modifications, and no additional training stages. Across multiple models, training regimes, and benchmarks, injecting only a small fraction (3–10%) of such visually grounded instructions consistently improves performance on vision-centric evaluations. Our findings highlight instruction tuning with visually grounded SSL tasks as a powerful lever for improving visual perception in MLLMs through simple adjustments to the training data distribution.
Project page

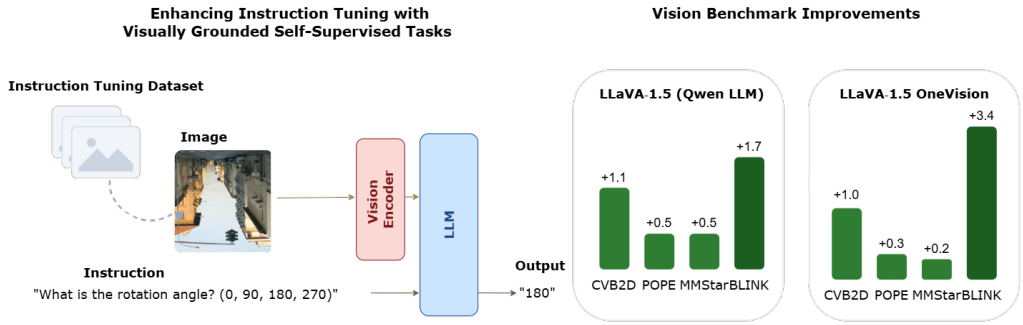


Figure 1: **V**isually **G**rounded **I**nstruction **F**ine-**T**uning V-GIFT. We enhance visual instruction tuning by injecting visually grounded self-supervised tasks as additional instruction-following examples sampled from the instruction-tuning data (left; rotation prediction shown). This simple modification encourages better use of visual information and yields consistent gains on vision-centric benchmarks (right; CVB-2D, POPE, MMStar, BLINK) across model variants.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Multimodal Large Language Models (MLLMs) [1, 41, 47] combine pretrained vision encoders with large language models (LLMs) [7, 64, 61] to perform multimodal instruction following. Modern open-source systems [48, 47, 40, 43] typically consist of three components: a pretrained vision encoder, a lightweight projection module that maps visual features into the language embedding space, and a pretrained LLM decoder. Trained via vision–language alignment followed by visual instruction tuning, these models achieve strong performance on image captioning, visual question answering, and general multimodal dialogue.

Despite their apparent competence, MLLMs frequently stumble on *vision-centric tasks* that require fine-grained visual understanding, such as object counting, spatial positioning, and geometric relation understanding [25, 62, 63]. While one might expect these failures to stem from weak visual representations [63], recent evidence suggests a more nuanced bottleneck: modern encoders such as CLIP [56] and DINOv2 [54] already capture rich, task-relevant features, yet the LLM component often under-utilizes this visual information during decoding [24]. This shift in understanding reframes the problem: the issue is not merely representational capacity, but training dynamics. During visual instruction tuning, models are optimized on image–instruction–response triplets expressed entirely in natural language. While flexible, this formulation introduces an unintended bias: many instruction-tuning examples can be partially or fully solved using strong language priors alone [20]. Besides, most of the times, these examples are automatically generated from generic captions of the images [48] without enforcing the use of vision cues for solving them. As a result, the model may learn language-dominant shortcut strategies and under-rely on visual evidence, even when vision is necessary. Notably, this limitation persists even as models scale and training data increases [20].

We hypothesize that this imbalance in supervision during instruction tuning is a central cause of weak vision-centric perception. If models are rarely required to depend on visual input in order to succeed, they tend to default to language-based heuristics. We propose to address this supervision imbalance within the visual instruction tuning phase. Our key idea is to reinterpret instruction tuning as a modality competition process: when many tasks can be solved through linguistic priors alone, training may implicitly favor language-dominant strategies. To counteract this bias, we inject inherently vision-forcing tasks into the instruction distribution. Specifically, we reformulate self-supervised (SSL) tasks, such as rotation prediction [29], color matching [78], and point correspondence [70], as image–instruction–response triplets compatible with standard MLLM pipelines. These tasks possess two critical properties: **(1)** they are visually grounded by construction, the correct answer cannot be inferred from language priors alone; and **(2)** they require no human annotation, as supervision is derived automatically from image transformations or feature-based correspondences.

Unlike recent works that incorporate self-supervised objectives through auxiliary losses or reinforcement learning with verifiable rewards (RLVR), our method does not modify the optimization paradigm. We retain standard autoregressive cross-entropy training and instead adjust the *distribution of instruction-following tasks* to systematically include vision-only problems that compel the model to rely on visual tokens, as language priors provide no information about a grayscale pixel's original color or a randomly rotated image's orientation. By integrating these tasks into visual instruction tuning, we encourage more effective coordination between visual perception and language reasoning without requiring auxiliary losses, architectural changes, or expensive RLVR pipelines.

Across multiple benchmarks, MLLM backbones, and training regimes, we observe consistent improvements in vision-centric perception. Remarkably, injecting a small fraction (between 3% and 10%) of visually grounded SSL instructions yields measurable gains across diverse benchmarks, and these improvements generalize to stronger models such as OneVision-1.5 [2]. Control experiments with matched training iterations confirm that the gains are not attributable to additional compute. We further show that SSL supervision is most effective when mixed directly into instruction tuning rather than applied as a separate pre- or post-training stage, highlighting the importance of shaping supervision during multimodal alignment. Finally, we find that even SSL tasks generated from a single image (through random view sampling) [3] can produce improvements, suggesting that the key factor is not dataset scale but the presence of objectives that compel visual grounding.

To summarize, our contributions are:

- We propose a simple yet effective framework to reformulate classic self-supervised pretext tasks as instruction-following data, mitigating language shortcuts in MLLMs and encouraging the LLM to better utilize visual representations.
- Our framework integrates seamlessly into the instruction-following pipeline. It is applicable to any MLLM architecture without architectural modifications and avoids additional pre- or post-training steps as well as complex hyperparameter tuning.
- Through extensive experiments, we show consistent improvements across models, training regimes, and benchmarks, while requiring minimal additional compute.

2 Related Work

Multimodal Large Language Models. MLLMs have emerged as a natural extension of LLMs to modalities beyond text [8], integrating non-textual information through two dominant architectural paradigms. In cross-attention-based models [1, 38], visual features are injected via dedicated cross-attention layers interleaved within the LLM. In projection-based models, exemplified by LLaVA [48, 47], a vision encoder, typically a CLIP-style [56, 65] encoder, maps images into visual embeddings with a lightweight adapter. The latter has become the dominant paradigm, with most modern MLLMs adopting this design for its simplicity and ease of training [47, 40, 2, 6, 69, 62, 60, 36]. LLaVA models follow a two-stage recipe: adapter pretraining on image-caption pairs, followed by visual instruction tuning of the full model. Recent improvements on the data front include introducing a mid training stage [2] to inject additional knowledge before instruction tuning; or curating high-quality human-annotated visually grounded instructions yielding richer supervision and stronger spatial grounding [19]. In this work we also take the data-centric approach to improve capabilities of MLLMs, yet we do not use any manual labels and leverage instead self-supervised pretext objectives.

Vision-centric Strategies for MLLMs. Improving visual perception in MLLMs focuses on two main axes. First line of work focuses on designing more expressive projectors [53, 47, 12], for example by aggregating multi-layer features from the vision encoder before being fed into the LLM [13, 44], while other studies explore the use of multiple vision encoders [63, 37, 62, 4, 58, 52]. Second line of work identifies visual bottleneck in utilization of visual information during LLM decoding [24]. Motivated by this insight, recent works introduce auxiliary objectives that directly supervise visual tokens within the LLM decoder. These include reconstruction-based losses applied to visual token outputs [67] and distillation of intermediate LLM features from external vision foundation models [77, 9], such as DINOv2 [54]. In contrast to these approaches, which modify model architectures or introduce auxiliary optimization objectives, we focus on instruction tuning and show that adjusting the supervision distribution through visually grounded self-supervised instructions is sufficient to improve visual information usage. As a result, our approach remains simple and lightweight: it preserves the standard MLLM training objective and architecture, with only marginal overhead from a small fraction of SSL-style instruction examples.

Leveraging Self-supervised Learning. Self-supervised learning (SSL) learns visual representations from unlabeled data through annotation-free pretext tasks ranging from low-level pretext tasks such as: predicting rotation angles [29], relative patch positions [21], colorization [78], to richer, high-level objectives including contrastive learning [34, 73, 16, 15], prototype-based clustering [10, 26], self-distillation [30, 11, 54, 28, 27, 66], and masked image modeling [33]. Recently, frameworks for improving visual utilization in MLLMs also build on SSL, e.g. masked image modeling [67], adapting jigsaw solving as an RLVR post-training [72, 71, 31, 49]. In this work, we similarly draw inspiration from SSL pretext tasks and highlight a visually grounded data imbalance in visual instruction tuning, which we address by directly incorporating self-supervised tasks into the existing instruction format, without introducing auxiliary losses, additional training stages, or costly RLVR pipelines.

3 Method

Our goal is to improve the performance of MLLMs on vision-heavy tasks by encouraging stronger reliance on visual information during training. Rather than modifying model architectures or introducing additional training stages, we intervene directly in the visual instruction tuning phase. Specifically, we augment standard instruction tuning with a small number of automatically-generated visually

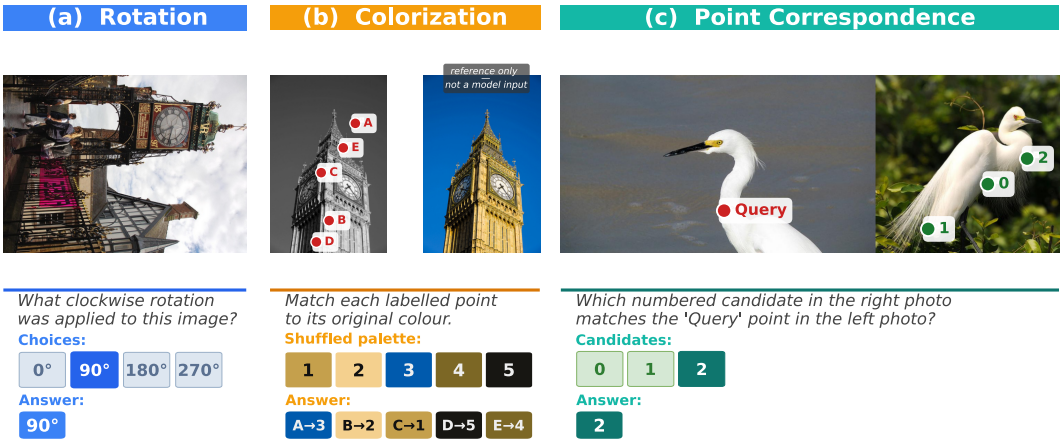


Figure 2: Visually grounded instruction-following tasks reformulated from self-supervised tasks. (a) **Rotation prediction:** the model must infer orientation from the image. (b) **Point-wise colorization:** the model must match grayscale points to their original colors, requiring fine-grained visual discrimination and spatial grounding. (c) **Point correspondence:** the model must identify corresponding points across views, requiring spatial reasoning. Overall, these tasks encourage reliance on visual evidence and integration of local and global visual cues, rather than language priors.

grounded tasks, formulated as natural language instructions, that require genuine visual reasoning and cannot be reliably solved using language priors alone.

3.1 Multimodal Large Language Model Training Pipeline

We adopt a standard MLLM architecture consisting of a pretrained vision encoder, a projector, and a pretrained LLM decoder. The vision encoder extracts visual features from the input image, which are mapped by the projector into the LLM's token embedding space and processed jointly with text tokens by the LLM. Training follows the commonly used two-stage pipeline employed by LLaVA-style models [48, 47, 40, 2]: vision–language alignment pretraining followed by visual instruction tuning.

Vision–language alignment. In the first stage, the vision encoder is frozen and the projection module is trained to align visual features with the LLM embedding space using large-scale image–caption pairs. Each training sample consists of an image I and an associated caption $T = (t_1, \ldots, t_N)$. The model is optimized with an autoregressive language modeling objective:

$$\mathcal{L}_{\text{align}} = -\sum_{i=1}^{N} \log p_\theta(t_i \mid t_{<i}, I). \tag{1}$$

This stage yields a base MLLM capable of jointly processing visual and textual inputs. Recently, OneVision-1.5 [2] extends the vision–language alignment stage with extensive mid-training (stage 1.5) of the entire model on high-quality image–text pair data.

Visual instruction tuning. In the second stage, either the full model (e.g., OneVision-1.5 [40, 2]) or only the projector with the LLM (e.g., LLaVA [48, 47]) are fine-tuned on multimodal instruction-following data. Each sample consists of an image I, a textual instruction $x = (x_1, \cdots, x_N)$, and a response $y = (y_1, \ldots, y_M)$. The model is trained to generate the response conditioned on both the instruction and the visual input using the standard autoregressive loss:

$$\mathcal{L}_{\text{inst}} = -\sum_{j=1}^{M} \log p_\theta(y_j \mid y_{<j}, x, I). \tag{2}$$

This stage shapes the model's instruction-following behavior and multimodal reasoning. `V-GIFT` operates exclusively within this instruction tuning phase.

3.2 Visual Self-Supervised Tasks as Instructions

We introduce a set of visual self-supervised learning (SSL) tasks that derive supervision directly from image structure. Unlike conventional SSL methods, which rely on auxiliary losses or specialized

heads during vision encoder pretraining, we reformulate these tasks as natural language instructions compatible with standard instruction tuning. This enables seamless integration into existing MLLM pipelines without architectural or optimization changes.

Each SSL task is expressed as an image–instruction–response triplet (I, x, y), where the instruction x specifies a visually grounded task and the response y is an answer automatically derived from the image. These tasks follow the same format as standard multimodal instruction data and are optimized using the autoregressive cross-entropy loss in Eq. 2. We consider three SSL tasks (Figure 2).

Rotation prediction. Given an image I, we generate a rotated version $\tilde{I} = R_\theta(I)$, where $\theta \in \{0°, 90°, 180°, 270°\}$ is sampled uniformly at random and independently of the image content. The instruction asks the model to identify the applied rotation and to answer only with the degree value. The response is the discrete label $y = \theta$, serialized as text. Each training example takes the form:

$$(\tilde{I},\ x = \text{“What is the rotation angle of this image?”},\ y = \text{“}\{\theta\}\text{”}). \tag{3}$$

We illustrate this rotation prediction instruction task in Figure 2 (a). Because the rotation label is independent of both the instruction and dataset statistics, it cannot be predicted without inspecting the visual input. Solving the task requires inferring object orientation from visual evidence in the image.

Point-wise colorization (color matching). Given a color image $I \in \mathbb{R}^{H\times W\times 3}$, we convert it to grayscale and sample K spatial points $\{q_i\}_{i=1}^{K}$. For each point, we compute its average RGB color c_i over a local $r \times r$ neighborhood and enforce diversity via $\|c_i - c_j\|_2 \geq \delta$, using rejection sampling, where δ is a fixed threshold. Each point is assigned a label (e.g., $A, B, \dots$) and overlaid on the grayscale image $\tilde{I}$.

We randomly permute the candidate colors and present them as a numbered list (RGB triplets with optional coarse color names). The instruction asks the model to match each labeled point to its corresponding color index, with responses formatted as label–index pairs:

$$(\tilde{I},\ x = \text{color-matching instruction},\ y = \text{“}A\text{-}\{y_A\}, B\text{-}\{y_B\}, \dots\text{”}), \tag{4}$$

where y_A is the index of the true color for point A.

We illustrate this task in Figure 2 (b). Although candidate colors are specified textually, their ordering is randomized and the correct point–color mapping is determined solely by the image. As a result, the task cannot be resolved from language priors alone. The model must localize each marked point and infer its color from visual evidence in the grayscale image, requiring fine-grained spatial grounding rather than semantic guessing.

Point correspondence. Given an image pair (I_1, I_2) depicting the same object instance, we ask the model to identify corresponding points across views. Following DIP [59], we automatically generate supervision using pseudo-segmentation masks obtained with Stable Diffusion [32] and dense DINOv2 [54] features. These signals restrict point sampling to object regions with consistent semantic identity across views and enable the identification of reliable correspondences.

Specifically, we sample a query point q in I_1, identify its best-matching location q^+ in I_2 via dense feature similarity, and sample two distractor points from the same object region. The candidates are randomly permuted and labeled $(0, 1, 2)$. The instruction asks which candidate corresponds to the query point, and the response is the index of the correct match:

$$((I_1, I_2),\ x = \text{correspondence instruction},\ y = \text{“}\{y_{q^+}\}\text{”}), \tag{5}$$

where $y_{q^+} \in (0, 1, 2)$ denotes the index of the best-matching location q^+ in I_2 among the three candidates. We illustrate this task in Figure 2 (c). Solving it requires identifying consistent visual features across viewpoints and reasoning about spatial correspondences between the two images. Note that our goal is not to propose a standalone self-supervised correspondence method, but to provide a simple automatically generated task for instruction tuning. We use pretrained vision models (e.g., Stable Diffusion masks and DINOv2 features) only offline to construct reliable correspondence signals from unlabeled images. These models are not involved during MLLM training, and we do not use feature distillation or auxiliary supervision.

Collectively, these tasks require the model to develop core visual reasoning abilities, including sensitivity to object geometry and orientation, fine-grained visual discrimination, precise spatial grounding, and cross-view correspondence. Solving them also requires integrating local visual cues with global image context while relying on visual evidence rather than language priors.

3.3 Integrating SSL Instructions into Instruction Tuning

We incorporate the proposed SSL tasks into visual instruction tuning by augmenting the original dataset with automatically generated SSL instruction samples. Let $\mathcal{D}_{\text{inst}}$ denote the original instruction dataset and $\mathcal{D}_{\text{ssl}}$ the SSL samples. The final training set is $\mathcal{D} = \mathcal{D}_{\text{inst}} \cup \mathcal{D}_{\text{ssl}}$, from which mini-batches are sampled uniformly.

We control the amount of SSL supervision via the ratio $\rho = 100 \times \frac{|\mathcal{D}_{\text{ssl}}|}{|\mathcal{D}_{\text{inst}}|}$. which defines the percentage of additional SSL samples relative to the original dataset. We treat ρ as a hyperparameter controlling the strength of visually grounded supervision. SSL samples can be generated from any of the tasks in Section 3.2, either individually or combined, providing complementary visual reasoning signals.

In practice, we use small ratios: $\rho = 10\%$ for LLaVA-1.5 and $\rho = 3\%$ for OneVision-1.5, reflecting differences in dataset scale and training dynamics. Despite their small proportion, these SSL samples consistently improve reliance on visual information and reduce language-driven shortcuts. Importantly, the additional computational cost scales linearly with ρ and remains negligible in all settings, requiring no changes to the optimization schedule, batch size, or learning rate.

4 Experiments

4.1 Experimental Setup

Models and training protocol. We build on the LLaVA-1.5 framework [47], using both Vicuna-7B-v1.5 [17] and Qwen2.5-7B [55] language model backbones and CLIP ViT-L/14 vision [56] encoder. We evaluate two training regimes: full model fine-tuning and parameter-efficient adaptation using LoRA [35]. To assess generalization beyond a single architecture and training recipe, we evaluate our method on the more recent OneVision-1.5 model [2]. Despite the related naming, OneVision-1.5 should be viewed as a distinct model family from LLaVA-1.5: it uses different architectural components, including a RICE-ViT vision encoder [75] and a Qwen3-4B language model [76], as well as a different training pipeline, image resolution, and instruction-tuning data. We train the visual instruction tuning stage on LLaVA-NeXT-780k, a subset of the OneVision-1.5-Instruct-Data [2], chosen for computational efficiency. We focus on these models as they provide open training recipes and instruction-tuning data, enabling modification of the instruction distribution. Other open-weight MLLMs (e.g., Qwen-VL [5]) do not release such data.

We augment instruction tuning with three SSL tasks (rotation, correspondence, colorization). Unless stated otherwise, $\rho = 10\%$ for LLaVA-1.5 (Vicuna, Qwen) and $\rho = 3\%$ for OneVision-1.5.

Training details. All experiments are trained on $4\times$H100 GPUs. Full fine-tuning on LLaVA-v1.5-mix665k of LLaVA-1.5-Qwen7B takes approximately 12 hours under this setup. Full fine-tuning on 110% of visual instruction following data of LLaVA-1.5-Qwen7B requires approximately 14 hours. We follow the original LLaVA-1.5 and OneVision-1.5 training recipes and keep all optimization hyperparameters identical to the standard instruction tuning configuration. For the main results in Table 1, we report mean performance and standard deviation over three independent training runs with different random seeds.

Evaluation benchmarks. We evaluate performance on several vision-centric multimodal benchmarks: CV-Bench 2D [62] (CVB-2D), POPE [42], MMStar [14], and BLINK [25], We additionally report results on generalist benchmarks MathVision [68], OCRBench [50] and RealWorldQA [74].

4.2 Main Results on Vision-Centric Benchmarks

As presented in Table 1 across almost all evaluation settings, incorporating SSL instruction tuning consistently improves performance over the baseline models. Gains are observed for both LLaVA-1.5-Vicuna and LLaVA-1.5-Qwen, which share the same vision encoder and training pipeline but differ in the LLM backbone, indicating that the effect is not specific to the decoder. Importantly, improvements also extend to OneVision-1.5, a stronger model with a distinct architecture, pipeline, and training data, demonstrating that our method generalizes beyond a single implementation and remains effective for more recent, higher-performing MLLMs.

Table 1: Main results of incorporating visually grounded SSL tasks on three MLLM models (LLaVA-1.5-Vicuna-7B, LLaVA-1.5-Qwen2.5-7B, OneVision-1.5-4B). Performance is reported on vision-centric and general benchmarks. Results are averaged over three independent training runs, with standard deviations reported as ±. Numbers in parentheses indicate improvement over the baseline.

	Vision-Centric					General		
Method	**CVB-2D**	**POPE**	**MMStar**	**BLINK**	**Avg.**	**MathVision**	**OCRB**	**RealWQA**
LLaVA-1.5-Vicuna-7B								
Baseline	55.9	87.0	33.5	**38.2**	53.6	12.9	319.3	54.2
V-GIFT	**58.5** (+2.6) ±1.52	**87.2** (+0.2) ±0.29	**34.6** (+1.1) ±0.25	37.8 (-0.4) ±0.13	**54.5** (+0.9) ±0.34	**13.3** (+0.4) ±1.24	318.3 (-1.0) ±3.06	**55.4** (+1.2) ±1.29
LLaVA-1.5-Qwen2.5-7B								
Baseline	61.1	87.2	42.7	40.1	57.8	12.2	312.0	56.4
V-GIFT	**62.2** (+1.1) ±1.53	**87.7** (+0.5) ±0.33	**43.2** (+0.5) ±0.64	**41.8** (+1.7) ±0.75	**58.7** (+0.9) ±0.64	**15.3** (+3.1) ±2.48	311.3 (-0.7) ±6.35	**59.0** (+2.6) ±0.40
OneVision-1.5-4B								
Baseline	70.0	88.6	55.3	48.8	65.7	18.69	627.0	66.1
V-GIFT	**70.9** (+0.9) ±0.96	**88.7** (+0.1) ±0.33	**55.4** (+0.1) ±0.52	**52.5** (+3.7) ±0.79	**66.9** (+1.2) ±0.25	17.37 (-1.32) ±0.09	**634.0** (+7.0) ±3.54	**66.4** (+0.3) ±0.09

Table 2: Effect of SSL tasks with LoRA training on LLaVA-1.5-Qwen2.5. Numbers in parentheses indicate improvement over the baseline. Baseline results are reproduced, while VIRAL results are reported from [77]. POPE* represents average accuracy across the "random" and "popular" subsets.

Model	**Method**	**CVB-2D**	**POPE***	**MMStar**	**BLINK**	**Avg.**
LLaVA-1.5-Qwen2.5 (LoRA)	Baseline	59.9	87.7	42.6	40.0	57.4
	V-GIFT	**63.8** (+3.9)	**88.5** (+0.8)	**43.7** (+1.1)	**43.3** (+3.3)	**59.6** (+2.2)
	VIRAL [77]	60.5	88.3	39.2	–	–

In Table 2, we evaluate our method on LLaVA-1.5-Qwen using LoRA fine-tuning [35], a parameter-efficient adaptation approach, instead of full fine-tuning. The results demonstrate substantial performance improvements even in this parameter-efficient setting. We also compare against VIRAL [77], a recent method that enhances vision-centric benchmarks via auxiliary distillation losses (reported in the LoRA setting). Despite its simplicity and the absence of additional objectives or architectural modifications, our approach achieves higher overall performance. These findings suggest that carefully adjusting the instruction tuning distribution can be more effective than introducing specialized loss functions for improving vision-centric instruction-following performance.

We report results on general benchmarks in Table 1. We observe improvements on MathVision and RealWorldQA for LLaVA-1.5-Qwen and on OCRBench for OneVision-1.5, on-par for LLaVA-1.5-Vicuna, indicating that injecting vision-centric SSL tasks does not hurt general reasoning skills.

4.3 Experimental Analysis

Impact of each SSL pretext task. In Table 3, we analyze the effect of incorporating each SSL instruction task individually, as well as their combination, during instruction tuning. Each SSL task individually improves average performance over the baseline, and combining all three yields further gains that are both stronger and more consistent across benchmarks. This suggests that the tasks provide complementary supervision signals, with their joint use producing the most robust improvements in vision-centric instruction-following. Rotation prediction and color matching alone yield consistent gains, indicating that improvements are not specific to the correspondence task.

How much SSL is enough? Impact of injection ratio ρ. In Figure 3 we evaluate the impact of the SSL injection ratio ρ, which controls the proportion of SSL instruction samples added to the instruction tuning dataset, for LLaVA-1.5-Qwen and OneVision-1.5. Although the two models exhibit different sensitivity to ρ, likely due to differences in dataset scale, architecture, and training pipeline, the overall trend is consistent. Even a small fraction of SSL data ($\rho = 1\%$) yields measurable improvements over the baseline ($\rho = 0\%$). Performance peaks at $\rho = 10\%$ for LLaVA-1.5-Qwen and $\rho = 3\%$ for OneVision-1.5, after which gains saturate or slightly decline. These results indicate that modest amounts of SSL-based visually grounded supervision are sufficient to improve vision-centric instruction-following performance, while larger proportions provide limited additional benefit. Since ρ is defined relative to the size of the instruction-tuning dataset, V-GIFT scales the amount of SSL

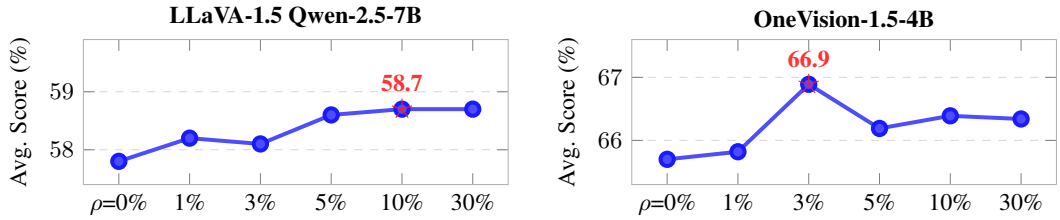


Figure 3: Effect of the SSL injection ratio ρ on vision-centric instruction-following performance for LLaVA-1.5-Qwen2.5-7B (left) and OneVision-1.5-4B (right).

Table 3: **SSL task ablation.** Impact of individual and combined SSL pretext tasks on LLaVA-1.5-Qwen2.5-7B ($\rho = 1\%$ per task).

SSL task			Benchmark				
Rot.	**Col.**	**Corr.**	**CVB-2D**	**POPE**	**MMStar**	**BLINK**	**Avg.**
			61.1	87.2	42.7	40.1	57.8
✓			**62.4**	87.2	42.5	40.7	58.2
	✓		62.3	87.5	41.8	41.7	58.3
		✓	61.9	87.3	41.7	**42.0**	58.2
✓	✓	✓	62.2	**87.7**	**43.2**	41.8	**58.7**

Table 4: **Training compute control.** Comparison against baselines trained with the same proportional increase in instruction-tuning iterations, but without SSL tasks, on OneVision-1.5-4B.

Training Setup	**CVB-2D**	**POPE**	**MMStar**	**BLINK**	**Avg.**
Baseline	70.0	88.6	55.3	48.8	65.7
Baseline (+3% instruct only)	68.2	**89.0**	**55.4**	49.3	65.6
V-GIFT (+3% SSL)	**70.9**	88.7	**55.4**	**52.5**	**66.9**

Table 5: **When to inject SSL.** Effect of SSL injection stage on LLaVA-1.5-Qwen2.5-7B. Training after instruction tuning is performed using LoRA to avoid catastrophic forgetting.

SSL Injection Stage	**CVB-2D**	**POPE**	**MMStar**	**BLINK**	**Avg.**
Baseline	61.1	87.2	42.7	40.1	57.8
SSL before IT (3%)	60.1	87.1	42.6	40.8	57.7
SSL before IT (100%)	57.6	87.6	43.9	39.1	57.1
SSL during IT (ours)	**62.2**	**87.7**	43.2	**41.8**	**58.7**
SSL after IT	36.5	50.2	39.5	33.2	39.9
SSL after IT (GRPO)	61.1	87.4	**43.5**	39.8	58.0

Table 6: **SSL image source.** Effect of SSL image source (rotation prediction, $\rho = 1\%$) on LLaVA-1.5-Qwen2.5-7B.

SSL Image Source	**CVB-2D**	**POPE**	**MMStar**	**BLINK**	**Avg.**
Baseline	61.1	87.2	42.7	40.1	57.8
Single image (augmented views)	**62.7**	86.9	42.5	**41.5**	**58.4**
COCO (reused images)	62.4	**87.2**	42.5	40.7	58.2

supervision with the instruction tuning data, maintaining a balance between general instruction-following and visually grounded tasks across different training regimes.

Are Gains Due to Additional Training Compute? Injecting SSL tasks with ratio ρ increases the total number of instruction tuning samples, resulting in a proportional increase in training iterations and compute. Although ρ is small in our setting, we verify that the observed improvements do not simply stem from additional training. In Table 4, we compare our method against a control baseline trained with the same increase in training iterations but using only standard instruction data, without SSL tasks. Specifically, we use the OneVision-1.5 framework and extend training by re-exposing the model to previously seen instruction data. The results show that additional training compute alone does not improve performance. Gains arise when the extra data consists of SSL tasks, confirming that improvements are driven by the nature of the supervision rather than the slightly longer training.

Where Should SSL Be Applied? Our method integrates SSL-based visually grounded tasks during the visual instruction tuning stage. In Table 5, we evaluate alternative injection strategies to understand whether timing matters. Specifically, we compare three settings: (a) adding an SSL-only stage before the original instruction tuning, (b) mixing SSL tasks during instruction tuning (our method), and (c) adding an SSL-only stage after the original instruction tuning.

We observe that improvements occur only when SSL tasks are injected during instruction tuning. Applying SSL before instruction tuning yields performance comparable to the baseline, even with increased data size (100%), suggesting that a subsequent instruction tuning phase largely overrides the effect of a standalone SSL stage. Injecting SSL after instruction tuning causes significant performance degradation due to catastrophic forgetting. In this case, even careful tuning of the SSL ratio ρ, reducing it from 10% to 1%, and using LoRA [35], cannot fully recover general instruction-following ability. Training with GRPO [57] avoids this collapse and even slightly improves POPE and MMStar over the baseline, but the gains are not consistent across benchmarks and the average score remains below SSL during instruction tuning. This suggests that while RL post-training can preserve general ability



Figure 4: Attention map from the Baseline (LLaVA-1.5-Vicuna-7B) and V-GIFT on CVB-2D examples. V-GIFT produces *more focused and better localized* attention on task-relevant objects.

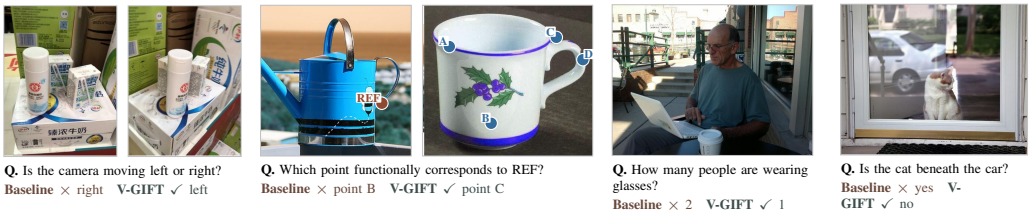


Figure 5: **Qualitative examples.** We present a few qualitative examples comparing LLaVA-1.5 Qwen-2.5-7B baseline against V-GIFT. Our SSL-inspired tasks yield improvements on the variety of vision oriented skills such as counting, multi-view reasoning and visual reasoning.

better than supervised finetuning, it does not replace the benefit of exposing the model to visually grounded instructions during the main instruction-tuning stage. Overall, these results indicate that integrating SSL supervision directly within instruction tuning is both more effective and more robust than introducing separate pre- or (RL-based) post-training SSL stages.

Effect of SSL Image Source. In Table 6, we analyze how the choice of image source for constructing SSL tasks influences vision-centric instruction-following performance, using rotation prediction as a representative SSL objective. We consider two contrasting settings: (i) COCO, which is already part of the original instruction tuning data and can be reused without introducing additional images, and (ii) a single high-resolution image [3], from which multiple augmented views are generated via geometric cropping and color jittering to create diverse SSL samples. Both sources lead to consistent improvements over the baseline on vision-centric benchmarks. Notably, even SSL supervision derived from multiple views of a single image provides gains. These findings suggest that the critical factor is not dataset scale or diversity, but the presence of visually grounded objectives that force the model to rely on visual input. In such tasks, language priors alone are insufficient, encouraging stronger alignment between visual tokens and the language model.

4.4 Analysis of Visual Information Utilization

V-GIFT reduces language priors. We follow a recent language priors quantization technique proposed in [51]. In Table 7 we present the mean TVI scores on CVBench-2D and MMStar benchmarks comparing baseline LLaVA-1.5 Vicuna 7B trained with standard Instruction Tuning dataset and the model trained with V-GIFT. We observe that model trained with our improved strategy achieves higher TVI scores on both datasets, indicating that the SSL-based auxiliary tasks reduce the model's reliance on language priors.

Semantic information in LLM representations. To investigate the effect of V-GIFT on internal visual representations, we evaluate the quality of visual representations encoded by the LLM using a linear semantic segmentation probe on PASCAL VOC [23], following [46]. We report the mIoU from the best-performing layer, layer 25 for LLaVA-1.5-Qwen2.5-7B and layer 11 for OneVision-1.5-4B; the same layers were optimal for both V-GIFT and the baseline. We report results averaged over three independent training runs. As shown in Table 8, V-GIFT yields improvements across both model architectures, despite the absence of segmentation supervision during training. These results

Table 7: Mean TVI ↑ [51] on LLaVA-1.5 Vicuna-7B.

Method	CVB-2D	MMStar
Baseline	0.1238	0.1426
V-GIFT	**0.1368**	**0.1430**

Table 8: Linear semantic segmentation probing on PASCAL VOC using visual representations encoded by the LLM. We report mIoU (%), averaged over three training runs.

Model	Method	mIoU	Δ
OneVision-1.5-4B	Baseline	33.9	–
	V-GIFT	**34.1**	+0.2
LLaVA-1.5-Qwen2.5-7B	Baseline	44.1	–
	V-GIFT	**44.5**	+0.4

Table 9: VLind-Bench for OneVision-1.5-4B: robustness to commonsense biases and language priors. Results are averaged over three runs.

Method	Acc_{CB}	Acc_{LP}
Baseline	58.2	43.0
V-GIFT	**58.5**	**43.6**

suggest that V-GIFT improves the quality of visual features learned by the LLM beyond task-specific adaptation.

Robustness to language priors. We further evaluate reliance on visual information using VLind-Bench [39], which measures the ability to overcome commonsense biases and language priors. Acc_{CB} and Acc_{LP} denote commonsense bias and language prior accuracy, respectively. As shown in Table 9, which reports averaged results over three training runs, V-GIFT improves performance for both commonsense biases and language priors. These results complement our TVI analysis, providing further evidence that visually grounded SSL instructions encourage greater reliance on visual evidence rather than language-based shortcuts.

V-GIFT improves attention to visual details. In Figure 4 we visualize attention maps on CVBench-2D, comparing where the baseline LLaVA-1.5-Vicuna and the one trained with V-GIFT focus attentions when processing visual inputs. We notice that the V-GIFT model concentrates attention more precisely on the relevant objects (the lamp and the television). This suggests that SSL-augmented instruction tuning encourages the model to ground its responses in localized visual evidence.

Qualitative examples on vision-centric benchmarks. We present in Figure 5 qualitative examples comparing LLaVA-1.5-Qwen (Baseline) against V-GIFT across visually demanding tasks, including counting, spatial relation understanding, visual reasoning, multi-view reasoning, and functional correspondence. In each case, the baseline fails by defaulting to plausible language-driven responses, while our model produces the correct answer by integrating visual information more effectively.

5 Conclusion

This paper introduces V-GIFT, a novel framework designed to leverage self-supervised learning within the visual instruction tuning pipeline. We instantiate V-GIFT using three diverse pretext tasks (image rotation, colorization, and point correspondences) which are integrated seamlessly without requiring architectural modifications nor modifications to training recipes. Our experiments show that V-GIFT yields consistent performance gains across a variety of vision-centric MLLM benchmarks while maintaining competitive generic reasoning capabilities and requiring minimal additional computational overhead. Furthermore, we provide qualitative analyses showing that our approach enables models to better ground their reasoning in fine-grained visual information. We hope V-GIFT will stimulate further research into the effective integration of low-level visual cues for complex multi-modal reasoning. Future work will explore extending our approach to other modalities, e.g., 3D point clouds or audio inputs.

Acknowledgments

This work was supported by the European Union's Horizon Europe research and innovation programme under grant agreement number 101214398 (ELLIOT), by HPC resources from GENCI-IDRIS (Grants AS011017181, AD011015037R2, AD011015037R1), and project RODEO (ANR-24-CE23-5886).

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *NeurIPS*, 2022. 2, 3

[2] Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Didi Zhu, et al. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*, 2025. 2, 3, 4, 6

[3] Yuki M Asano, Christian Rupprecht, and Andrea Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. *ICLR*, 2020. 2, 9, 16

[4] Mozhgan Nasr Azadani, James Riddell, Sean Sedwards, and Krzysztof Czarnecki. Leo: Boosting mixture of vision encoders for multimodal large language models. *arXiv preprint arXiv:2501.06986*, 2025. 3

[5] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023. 6

[6] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. 3

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 2

[8] Davide Caffagni, Federico Cocchi, Luca Barsellotti, Nicholas Moratelli, Sara Sarto, Lorenzo Baraldi, Marcella Cornia, and Rita Cucchiara. The revolution of multimodal large language models: A survey. In *ACL*, 2024. 3

[9] Davide Caffagni, Sara Sarto, Marcella Cornia, Lorenzo Baraldi, Pier Luigi Dovesi, Shaghayegh Roohi, Mark Granroth-Wilding, and Rita Cucchiara. Seeing beyond words: Self-supervised visual learning for multimodal large language models. *arXiv preprint arXiv:2512.15885*, 2025. 3

[10] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 2020. 3

[11] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 3

[12] Junbum Cha, Wooyoung Kang, Jonghwan Mun, and Byungseok Roh. Honeybee: Locality-enhanced projector for multimodal llm. In *CVPR*, 2024. 3

[13] Gongwei Chen, Leyang Shen, Rui Shao, Xiang Deng, and Liqiang Nie. Lion: Empowering multimodal large language model with dual-level visual knowledge. In *CVPR*, 2024. 3

[14] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *NeurIPS*, 2024. 6, 17

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 3

[16] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 3

[17] Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023. 6

[18] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *ICLR*, 2024. 15, 16

[19] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *CVPR*, 2025. 3

[20] Ailin Deng, Tri Cao, Zhirui Chen, and Bryan Hooi. Words or vision: Do vision-language models have blind faith in text? In *CVPR*, 2025. 2

[21] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *CVPR*, 2015. 3

[22] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *ACMMM*, 2024. 16

[23] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 9

[24] Stephanie Fu, Tyler Bonnen, Devin Guillory, and Trevor Darrell. Hidden in plain sight: Vlms overlook their visual representations. *arXiv preprint arXiv:2506.08008*, 2025. 2, 3

[25] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *ECCV*, 2024. 2, 6, 17

[26] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. In *CVPR*, 2020. 3

[27] Spyros Gidaris, Andrei Bursuc, Gilles Puy, Nikos Komodakis, Matthieu Cord, and Patrick Pérez. Obow: Online bag-of-visual-words generation for self-supervised learning. In *CVPR*, 2021. 3

[28] Spyros Gidaris, Andrei Bursuc, Oriane Siméoni, Antonín Vobeckỳ, Nikos Komodakis, Matthieu Cord, and Patrick Perez. Moca: Self-supervised representation learning by predicting masked online codebook assignments. *TMLR*, 2024. 3

[29] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2, 3

[30] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*, 2020. 3

[31] Xiaojun Guo, Runyu Zhou, Yifei Wang, Qi Zhang, Chenheng Zhang, Stefanie Jegelka, Xiaohan Wang, Jiajun Chai, Guojun Yin, Wei Lin, et al. Ssl4rl: Revisiting self-supervised learning as intrinsic reward for visual-language reasoning. *arXiv preprint arXiv:2510.16416*, 2025. 3

[32] Yatharth Gupta, Vishnu V Jaddipal, Harish Prabhala, Sayak Paul, and Patrick Von Platen. Progressive knowledge distillation of stable diffusion xl using layer level loss. *arXiv preprint arXiv:2401.02677*, 2024. 5

[33] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 3

[34] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 3

[35] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 2022. 6, 7, 8

[36] Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. 3

[37] Oğuzhan Fatih Kar, Alessio Tonioni, Petra Poklukar, Achin Kulshrestha, Amir Zamir, and Federico Tombari. Brave: Broadening the visual encoding of vision-language models. In *ECCV*, 2024. 3

[38] Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander Rush, Douwe Kiela, et al. Obelics: An open web-scale filtered dataset of interleaved image-text documents. *NeurIPS*, 2023. 3

[39] Kang-il Lee, Minbeom Kim, Seunghyun Yoon, Minsung Kim, Dongryeol Lee, Hyukhun Koh, and Kyomin Jung. Vlind-bench: Measuring language priors in large vision-language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4129–4144, 2025. 10

[40] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024. 2, 3, 4

[41] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023. 2

[42] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *EMNLP*, 2023. 6, 17

[43] Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. Vila: On pre-training for visual language models. In *CVPR*, 2024. 2

[44] Junyan Lin, Haoran Chen, Yue Fan, Yingqi Fan, Xin Jin, Hui Su, Jinlan Fu, and Xiaoyu Shen. Multi-layer visual feature fusion in multimodal llms: Methods, analysis, and best practices. In *CVPR*, 2025. 3

[45] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 15, 16

[46] Benlin Liu, Amita Kamath, Madeleine Grunde-McLaughlin, Winson Han, and Ranjay Krishna. Visual representations inside the language model. *arXiv preprint arXiv:2510.04819*, 2025. 9

[47] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *CVPR*, 2024. 2, 3, 4, 6

[48] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 2, 3, 4

[49] Yuhong Liu, Beichen Zhang, Yuhang Zang, Yuhang Cao, Long Xing, Xiaoyi Dong, Haodong Duan, Dahua Lin, and Jiaqi Wang. Spatial-ssrl: Enhancing spatial understanding via self-supervised reinforcement learning. *arXiv preprint arXiv:2510.27606*, 2025. 3

[50] Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *SCIS*, 2024. 6, 18

[51] Lin Long, Changdae Oh, Seongheon Park, and Sharon Li. Understanding language prior of lvlms by contrasting chain-of-embedding. *ICLR*, 2026. 9, 10

[52] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, et al. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*, 2024. 3

[53] Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Anton Belyi, et al. Mm1: methods, analysis and insights from multimodal llm pre-training. In *ECCV*, 2024. 3

[54] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 2, 3, 5

[55] A Yang Qwen, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengpeng Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint*, 2024. 6

[56] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2, 3, 6

[57] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 8

[58] Min Shi, Fuxiao Liu, Shihao Wang, Shijia Liao, Subhashree Radhakrishnan, Yilin Zhao, De-An Huang, Hongxu Yin, Karan Sapra, Yaser Yacoob, et al. Eagle: Exploring the design space for multimodal llms with mixture of encoders. *arXiv preprint arXiv:2408.15998*, 2024. 3

[59] Sophia Sirko-Galouchenko, Spyros Gidaris, Antonin Vobecky, Andrei Bursuc, and Nicolas Thome. Dip: Unsupervised dense in-context post-training of visual representations. In *ICCV*, 2025. 5, 15

[60] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, et al. Paligemma 2: A family of versatile vlms for transfer. *arXiv preprint arXiv:2412.03555*, 2024. 3

[61] Qwen Team et al. Qwen2 Technical Report. *arXiv preprint arXiv:2407.10671*, 2024. 2

[62] 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. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *NeurIPS*, 2024. 2, 3, 6, 16, 18

[63] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *CVPR*, 2024. 2, 3

[64] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. arxiv 2023. *arXiv preprint arXiv:2302.13971*, 2023. 2

[65] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025. 3

[66] Shashanka Venkataramanan, Valentinos Pariza, Mohammadreza Salehi, Lukas Knobel, Spyros Gidaris, Elias Ramzi, Andrei Bursuc, and Yuki M Asano. Franca: Nested matryoshka clustering for scalable visual representation learning. *arXiv preprint arXiv:2507.14137*, 2025. 3

[67] Haochen Wang, Anlin Zheng, Yucheng Zhao, Tiancai Wang, Zheng Ge, Xiangyu Zhang, and Zhaoxiang Zhang. Reconstructive visual instruction tuning. In *ICLR*, 2025. 3

[68] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024. 6, 17

[69] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025. 3

[70] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *CVPR*, 2019. 2

[71] Zifu Wang, Junyi Zhu, Bo Tang, Zhiyu Li, Feiyu Xiong, Jiaqian Yu, and Matthew B. Blaschko. Jigsaw-r1: A study of rule-based visual reinforcement learning with jigsaw puzzles. *TMLR*, 2025. 3

[72] Penghao Wu, Yushan Zhang, Haiwen Diao, Bo Li, Lewei Lu, and Ziwei Liu. Visual jigsaw post-training improves MLLMs. In *ICLR*, 2026. 3

[73] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018. 3

[74] xAI. Grok. 2024. 6, 18

[75] Yin Xie, Kaicheng Yang, Xiang An, Kun Wu, Yongle Zhao, Weimo Deng, Zimin Ran, Yumeng Wang, Ziyong Feng, Roy Miles, et al. Region-based cluster discrimination for visual representation learning. In *ICCV*, 2025. 6

[76] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 6

[77] Heeji Yoon, Jaewoo Jung, Junwan Kim, Hyungyu Choi, Heeseong Shin, Sangbeom Lim, Honggyu An, Chaehyun Kim, Jisang Han, Donghyun Kim, et al. Visual representation alignment for multimodal large language models. *arXiv preprint arXiv:2509.07979*, 2025. 3, 7, 18

[78] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016. 2, 3

A Limitations

Our current instantiation of V-GIFT explores three SSL-style tasks: rotation prediction, point-wise color matching, and point correspondence. While these tasks provide complementary visually grounded supervision, they cover only a subset of abilities that may benefit MLLMs. A broader exploration of SSL-inspired instruction tasks can be beneficial for tasks involving other modalities, 3D point clouds or audio inputs.

Our experiments are conducted on LLaVA-1.5 variants and OneVision-1.5, which provide sufficiently open training recipes and instruction-tuning data for modifying the instruction distribution. Extending the study to other strong open-weight MLLMs remains challenging when the corresponding instruction-tuning data are not fully released.

The optimal SSL injection ratio may need to be adjusted for different model families or data scales. The additional computational cost scales linearly with the amount of SSL data added; however in our experiments this overhead is small because we use low injection ratios.

B Broader Impacts

V-GIFT aims to improve visual grounding and reduce reliance on language priors in MLLMs, which may benefit applications requiring more reliable visual understanding across domains such as education, healthcare, and communication. At the same time, stronger visual reasoning capabilities may also improve systems used in sensitive or harmful downstream applications. It is important to remain mindful of potential risks, downstream use of improved MLLMs should be accompanied by appropriate safeguards, evaluation for bias and robustness, and application-specific risk assessment.

C Implementation Details

C.1 Self-supervised instruction-tuning tasks

C.1.1 Colorization task.

We construct a colorization-based visual reasoning task from the COCO 2017 training split [45], discarding grayscale images. For each image, we sample (N=5) points, each located at least 20 pixels from the image boundary. The color associated with the i-th point is defined as the mean RGB value $c_i \in \mathbb{R}^3$ computed over a $r \times r = 5 \times 5$ neighborhood centered at that point.

To avoid ambiguity, we enforce pairwise distinct colors among the five sampled points by requiring a minimum Euclidean distance of $\delta = 40$ in RGB space, i.e., $|c_i - c_j|_2 \geq \delta$ for $i \neq j$, implemented via rejection sampling. Each RGB value is mapped to a human-readable color name using the XKCD color vocabulary[1], using nearest-neighbor retrieval in RGB space.

The image is then converted to grayscale, and the sampled locations are annotated with labeled markers (e.g., $A, B, \ldots, E$), rendered as filled red circles with text labels. The five ground-truth colors are randomly shuffled and presented to the model as a numbered list in the format `RGB(r, g, b) (color name)`. The model must recover the correspondence between point labels and shuffled colors and output the mapping in the format "A-$\{y_A\}$, B-$\{y_B\}$, ...", where y_A denotes the index of the true color for point A. Example tasks are shown in Figure 6 (a).

C.1.2 Point correspondence task.

We construct a point correspondence task from a subset of paired images in the COCO 2017 training split using precomputed self-supervised segmentation masks and a fixed list of image pairs from [59]. For details on pseudo-segmentation mask extraction, we refer the reader to [59].

Given a pair of images $I_1, I_2 \in \mathbb{R}^{H \times W \times 3}$ and their corresponding pseudo-object segmentation masks $M_1, M_2 \in \{0, 1\}^{H \times W \times K}$, where K denotes the number of pseudo-classes (obtained in [59] via K-means clustering of DINOv2-ViT-B/14 [18] features), we first select an object of interest.

[1] The XKCD color vocabulary: https://xkcd.com/color/rgb/

Specifically, for each pseudo-class $k \in \{1, \ldots, K\}$, we compute the union of the corresponding regions across both masks, $M_1^k \cup M_2^k$, and select the class with the largest pixel area:

$$k^* = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} |M_1^k \cup M_2^k|$$

where $|\cdot|$ denotes pixel cardinality. The selected pseudo-label k^* defines the object of interest for the pair.

We then extract dense visual features using DINOv2-ViT-B/14 [18]. Specifically, we use the patch tokens from the final layer, yielding feature maps $A_1, A_2 \in \mathbb{R}^{h \times w \times 768}$ for the two images. A query point q is sampled uniformly from the selected region k^* in the first image such that $M_1^{k^*}(q) = 1$. Its corresponding patch feature $A_1(q) \in \mathbb{R}^{768}$ is matched against all patch features in the second image using cosine similarity $\operatorname{sim}(A_1(q), A_2(j)) = \frac{A_1(q) \cdot A_2(j)}{||A_1(q)||, ||A_2(j)||}$. The corresponding point q^+ is defined as the patch within the selected region k^* in I_2 with the highest similarity:

$$q^+ = \underset{j \in M_2^{k^*}}{\operatorname{argmax}} \operatorname{sim}(A_1(q), A_2(j)),$$

where, by abuse of notation, $j \in M_2^{k^*}$ indicates that the candidate patch j is restricted to the region corresponding to pseudo-class k^*. The final position of q^+ is taken as the center of the selected patch.

To form a 3-way multiple-choice task, we additionally randomly sample two distractor points from the same region $M_2^{k^*}$ in the second image. The three candidate points are then randomly shuffled and labeled $(0, 1, 2)$.

For single-image MLLMs, such as LLaVA-1.5, each example is rendered as a side-by-side composite of I_1 (left) and I_2 (right), with the query point shown in the left image and the candidate points in the right image. For models that accept multiple images, such as OneVision-1.5, I_1 and I_2 are fed separately. In both cases, points are visualized as red circles with text labels, and the model is asked to identify which candidate point in the second image corresponds to the query point on the shared object in the first image. Example tasks are shown in Figure 6 (b).

C.1.3 Rotation prediction task.

We construct a rotation prediction task using images from the COCO 2017 training split [45]. Each image is rotated clockwise by one of four discrete angles, $\theta \in \{0°, 90°, 180°, 270°\}$. The task is formulated as direct rotation prediction (in degrees): given a single input image, the model receives a fixed prompt stating that the image may be rotated by a multiple of $90°$ clockwise and must respond with one of $\{0, 90, 180, 270\}$. The target output is therefore the integer string corresponding to θ. Example tasks are shown in Figure 6 (c).

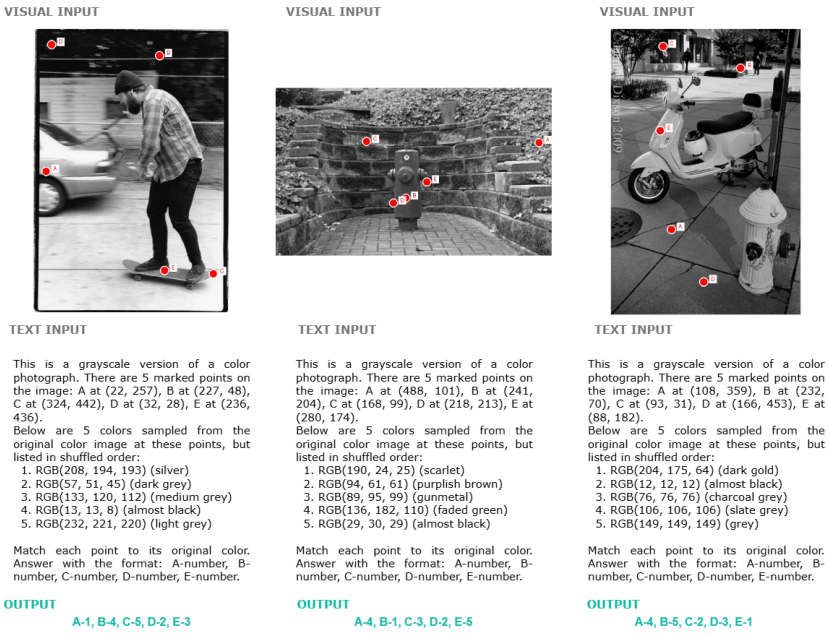
C.2 Single-image training data construction

In Table 6 of the main paper (subsection 4.3), we evaluate the impact of our method when the images used to construct the self-supervised instruction tuning tasks are derived from a single high-resolution image [3]. To obtain visual diversity despite the single-image setting, we generate a large set of augmented views using stochastic cropping and appearance transformations.

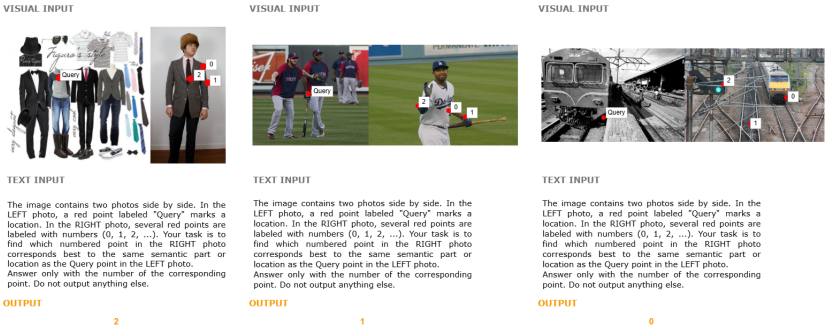
Specifically, each sample is produced by applying a random resized crop with crop area uniformly sampled from $[0.1\%, 8\%]$ of the original image area and aspect ratio sampled from $[3/4, 4/3]$, followed by resizing to 224×224 using bilinear interpolation. We then apply horizontal flipping with probability 0.5 and independent random adjustments of brightness, contrast, saturation, and hue. Brightness and contrast factors are sampled uniformly from $[0.75, 1.25]$, saturation from $[0.70, 1.40]$, and hue from $[-0.05, 0.05]$.

C.3 Evaluation protocols

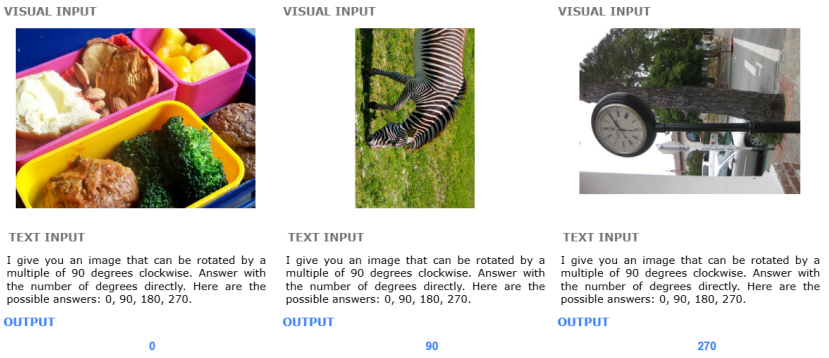
All evaluations are conducted using VLMEvalKit [22]. We do not rely on external API-based models; instead, we use the toolkit's built-in answer extraction and matching procedures. We report CVBench-2D [62] accuracy, computed over 1,438 2D spatial reasoning questions spanning the Count



(a) Colorization point-matching task.



(b) Point correspondence task.



(c) Rotation prediction task.

Figure 6: Examples of the visually grounded self-supervised tasks used during training: colorization point matching, point correspondence, and rotation prediction.

and Relation sub-tasks; POPE [42] mean accuracy, computed as the average accuracy across the random, popular, and adversarial COCO-POPE splits; MMStar [14], overall accuracy measured over 1,500 questions spanning six visual reasoning dimensions; and BLINK [25], overall accuracy measured across 14 visual perception sub-tasks. For general benchmarks, MathVision [68] is reported

Table 10: Effect of the SSL injection ratio ρ on vision-centric instruction-following performance for OneVision-1.5-4B.

Injection ratio ρ	CVB-2D	POPE	MMStar	BLINK	Avg.
1%	69.4	88.5	55.5	49.9	65.8
3%	**71.0**	**88.9**	**55.5**	52.2	**66.9**
5%	69.4	88.3	54.7	52.3	66.2
10%	69.5	88.3	54.1	**53.7**	66.4
30%	68.9	88.2	55.3	52.9	66.3

Table 11: Training hyperparameters for LLaVA-1.5 experiments.

Setting	LLaVA-1.5-Qwen2.5-7B Full	LLaVA-1.5-Vicuna-7B Full	LLaVA-1.5-Qwen2.5-7B LoRA
LLM backbone	Qwen2.5-7B-Instruct	Vicuna-7B-v1.5	Qwen2.5-7B-Instruct
Vision encoder	CLIP ViT-L/14-336	CLIP ViT-L/14-336	CLIP ViT-L/14-336
Training regime	Full fine-tuning	Full fine-tuning	LoRA fine-tuning
Epochs	1	1	1
Precision	bf16	bf16	bf16
Per-device batch size	1	8	1
Gradient accumulation steps	32	4	32
Number of GPUs	4	4	4
Global batch size	128	128	128
Learning rate	2×10^{-5}	2×10^{-5}	2×10^{-4}
MM projector learning rate	—	—	2×10^{-5}
Warmup ratio	0.03	0.03	0.03
LR scheduler	cosine	cosine	cosine
Max sequence length	2048	2048	2048
LoRA rank r	—	—	128
LoRA α	—	—	256

as accuracy over 3,040 questions spanning 16 mathematical sub-fields, OCRBench [50] is reported using the normalized final score, defined as the raw number of correct OCR sub-task answers divided by 10 to obtain a 0–100 scale, and RealWorldQA [74] is reported as overall accuracy across all questions. Most results reported in the paper are averaged over three independent training runs.

C.4 Attention map visualization

In Figure 4 of the main paper we present attention maps to visual tokens for random samples of CV-Bench2D [62]. Specifically, we visualize attention maps of the last token of the input sequence (which corresponds to the last token of the instruction) to all visual tokens in LLaVA-1.5 Vicuna 7B for the baseline and `V-GIFT`. Following [77] we measure the entropy of the attention weights and select the layer with the lowest spatial entropy, hence the strongest focus. We plot an average attention map of this layer through averaging the attention maps of all heads. We notice that `V-GIFT` yields more accurate attentions towards question-relevant objects compared to the baseline model.

C.5 Training Hyperparameters

We provide the main training hyperparameters used for the LLaVA-1.5 and OneVision-1.5 experiments in Table 11 and Table 12, respectively. We keep the optimization settings identical to the original instruction-tuning recipes and only modify the instruction-tuning data mixture.

D Additional Results

Detailed results for ρ sensitivity study. In Table 10 we present detailed result of the study of ρ parameter which complements Figure 3 of the main paper. In Table 13 we present a study of ρ parameter influence on general capabilities. Raising the injection ratio does not systematically degrade general capabilities. MathVision rises from 12.2 to 19.0 at 30%; RealWorldQA stays above baseline at every ratio and reaches 59.1 at 30%; OCRBench fluctuates around the baseline with no clear dependence on the ratio.

Table 12: Training hyperparameters for OneVision-1.5-4B instruction tuning.

Setting	OneVision-1.5-4B Full
LLM backbone	Qwen3-4B
Vision encoder	RICE-ViT, ViT-L/14
Training regime	Full fine-tuning
Trainable modules	LLM + adapter + vision encoder
Training steps	3,500
Precision	bf16
Global batch size	224
Learning rate	1×10^{-5}
Min. learning rate	1×10^{-6}
Warmup fraction	0.002
LR scheduler	cosine
Max sequence length	32,768
Max image tokens	1,000
Max image pixels	2,560,000
Optimizer	Adam (β_1=0.9, β_2=0.99, ε=10^{-5})
Gradient clipping	1.0

Table 13: Effect of the SSL injection ratio ρ on general capabilities for LLaVA-1.5-Qwen2.5-7B. Results are reported as mean $\pm$ standard deviation over three training runs.

Injection ratio ρ	MathVision	OCRBench	RealWorldQA
0% (baseline)	12.2 ± 2.1	312.0 ± 4.0	56.4 ± 0.3
1%	13.5 ± 1.2	313.3 ± 4.7	58.9 ± 0.5
3%	13.5 ± 1.2	**317.7** ± 3.8	58.0 ± 0.5
5%	15.3 ± 1.2	310.3 ± 1.2	57.0 ± 0.7
10%	15.3 ± 2.5	311.3 ± 6.4	59.0 ± 0.4
30%	**19.0** ± 1.0	313.3 ± 1.2	**59.1** ± 0.8

E Reliability of Self-Supervised Targets

The self-supervised tasks used in V-GIFT differ in how their training targets are obtained and, consequently, in their susceptibility to label noise. We describe the target generation procedures and assess their reliability below.

- **rotation prediction**. The targets are exact by construction, as the applied rotation angle is sampled from $\{0°, 90°, 180°, 270°\}$.
- **point-wise colorization**. The task is formulated as a multiple-choice matching problem rather than open-ended color prediction. The input image is converted to grayscale, labeled points are overlaid, and the model matches each point to its original color from a shuffled candidate list. To avoid ambiguous candidates, we enforce that the colors sampled at the five points are sufficiently separated: all pairwise Euclidean distances in RGB space must satisfy $\delta \geq 40$. This substantially reduces confusion between visually similar candidate colors. The targets are obtained directly from the original image pixels.
- **point correspondence**. Unlike the previous tasks, point correspondence relies on pseudo-labels derived from DINOv2 features and pseudo-segmentation masks, which may introduce noise. To quantify the extent of this noise, we conducted a manual audit of 50 correspondence examples drawn uniformly at random from the training set. Five annotators independently assessed whether each provided correspondence target was semantically sensible, and final labels were determined by majority vote. Under this protocol, 41 of the 50 examples (82.0%) were judged meaningful, suggesting that most pseudo-labels capture plausible semantic correspondences.

The single-task ablation in Table 3, provides a perspective on the contribution of individual tasks. Each task independently improves the average vision-centric score relative to the baseline, including point correspondence despite its imperfect pseudo-labels. The gains from point correspondence

also extend to benchmarks beyond correspondence-related tasks. These results show that tasks with different target generation mechanisms and levels of supervision reliability can each contribute to improved visual representations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are consistent with the paper's actual contributions. The stated claims are supported by the method and experiments across multiple tasks, models, training regimes, and vision-centric benchmarks in Sec. 4.2

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Sec. A

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not introduce theoretical results, theorems, or formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper describes the model families, training regimes, instruction-tuning datasets, SSL injection ratios, optimization protocol, compute setup, and evaluation benchmarks in Sec. 4.1. The SSL task construction and how these examples are added to the instruction-tuning data are described in Sec. 3.2, with additional implementation details in Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code: https://github.com/sirkosophia/V-GIFT. Models: https://huggingface.co/SophiaSirko/V-GIFT_llava_v1.5_qwen2.5_7B and https://huggingface.co/SophiaSirko/V-GIFT_llava_v1.5_vicuna7b.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup is described in Sec. 4.1, including the evaluated model families, training regimes, instruction-tuning datasets, SSL injection ratios, compute setup, and evaluation benchmarks. Additional SSL task construction and implementation details are provided in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The results are reported as the mean over three independent training runs with different random seeds, as described in Sec. 4.1. For the main table, we report the standard deviation across these runs to quantify variability.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer:[Yes]

   Justification: The paper reports the main compute resources in Sec. 4.1. It also discusses that the additional compute introduced by `V-GIFT` scales linearly with the SSL injection ratio in Sec. 3.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and believe the research conforms to it. The work uses standard public benchmarks and open MLLM training resources, does not involve human subjects or sensitive personal data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses broader impacts in Sec. B, including potential positive effects from improving visual grounding in MLLMs and potential risks.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a new high-risk pretrained model, image generator, or scraped dataset. The work modifies the instruction-tuning data distribution using automatically generated SSL-style examples built from existing datasets

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All models, datasets and code used are credited by citation, and we use them under their original licenses and terms of use for research purposes.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The primary new asset is the implementation code for the proposed training as well as implementation code for the creation of SSL tasks, which we released.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer:[N/A]

Justification: This paper does not involve research with human subjects, therefore IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer:[N/A]

Justification: The core method does not use an external LLM as an important, original, or non-standard component. The work studies and fine-tunes MLLMs using automatically generated visually grounded SSL-style instruction examples; any LLM use for writing, editing, or formatting does not affect the methodology, scientific rigor, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.