
Multi-Step Visual Reasoning with Visual Tokens

Scaling and Verification

Tianyi Bai^{1,3*}, Zengjie Hu^{2*}, Fupeng Sun^{4*}, Jiantao Qiu^{3†}, Yizhen Jiang²,
Guangxin He¹, Bohan Zeng², Conghui He^{3‡}, Binhang Yuan^{1‡}, Wentao Zhang^{2‡}

¹The Hong Kong University of Science and Technology, ²Peking University

³Shanghai Artificial Intelligence Laboratory, ⁴Imperial College London

hecongahui@pjlab.org.cn, biyuan@ust.hk, wentao.zhang@pku.edu.cn

Abstract

Multi-modal large language models (MLLMs) have achieved remarkable capabilities by integrating visual perception with language understanding, enabling applications such as image-grounded dialogue, visual question answering, and scientific analysis. However, most MLLMs adopt a static inference paradigm, encoding the entire image into fixed visual tokens upfront, which limits their ability to iteratively refine understanding or adapt to context during inference. This contrasts sharply with human perception, which is dynamic, selective, and feedback-driven. In this work, we introduce a novel framework for inference-time visual token scaling that enables MLLMs to perform iterative, verifier-guided reasoning over visual content. We formulate the problem as a Markov Decision Process, involving a reasoner that proposes visual actions and a verifier—trained via multi-step Direct Preference Optimization (DPO)—that evaluates these actions and determines when reasoning should terminate. To support this, we present a new dataset, VTS, comprising supervised reasoning trajectories (VTS-SFT) and preference-labeled reasoning comparisons (VTS-DPO). Our method significantly outperforms existing approaches across diverse visual reasoning benchmarks, offering not only improved accuracy but also more interpretable and grounded reasoning processes. These results demonstrate the promise of dynamic inference mechanisms for enabling fine-grained, context-aware visual reasoning in next-generation MLLMs. Code and datasets are publicly released at <https://vts-v.github.io/>.

1 Introduction

Multi-modal large language models (MLLMs) that can perceive and reason over visual content are a foundational component of modern AI systems. By extending large language models (LLMs) with visual perception capabilities, MLLMs support a wide range of applications—from image-grounded dialogue and visual question answering to robotics and scientific analysis. Yet despite their impressive generalization, one fundamental challenge remains unsolved: *how can we conduct effective inference-time scaling for MLLMs to enable fine-grained, context-aware interaction with visual information?*

Current MLLMs typically adopt a *static inference paradigm*—processing the whole image into a static fixed set of visual tokens in a single step, and conducting all reasoning based solely on this static embedding. This approach limits the model’s ability to recover from ambiguity, occlusion, or missing

*Equal contribution.

†Project leader.

‡Corresponding authors.



Figure 1: **Iterative Visual Reasoning with VTS-V.** Our framework equips both open-source and closed-source models with dynamic visual token scaling and step-wise verification to solve complex visual tasks. The example shows how VTS-V: (1) decomposes questions into executable steps, (2) invokes vision tools, and (3) iteratively refines answers via verifier feedback, achieving correct results. In contrast, vanilla models fail to ground detailed visual operations without token scaling, leading to incorrect answers.

detail: once the initial representation is formed, there is no mechanism to query the image again or refine visual understanding. In contrast, *human perception is inherently dynamic*—we iteratively inspect regions, zoom in, and seek new visual evidence as reasoning unfolds. Bridging this gap between static MLLM inference and dynamic human reasoning is the central problem addressed in this work.

Multi-step visual reasoning is critical for robust AI systems. Many tasks require identifying small objects, interpreting text in images, or reasoning about spatial relations—activities that benefit from an iterative, context-sensitive exploration of visual content. Furthermore, dynamic scaling enables more efficient reasoning by focusing computational resources on the most relevant parts of the image, rather than uniformly processing all pixels or tokens. Without this flexibility, existing models exhibit degraded performance on benchmarks such as BLINK [6], V-Star Bench [27], and MMVP [23], which are designed to probe deeper visual understanding.

The core technical challenge lies in the absence of an expressive framework for flexible, inference-time visual exploration. Existing approaches either construct improved visual reasoning datasets for fine-tuning [7, 21], which still rely heavily on the pretrained model’s image-text alignment quality, or attempt to enhance inference through text token scaling [2] and the use of external vision tools [11, 8, 22, 27]. However, these methods are limited in scope, focusing narrowly on static token expansion or employing a small set of fixed visual tools in predefined pipelines. As a result, these methods fail to give the model agency in choosing what to observe next, how to focus visual attention, or when to stop reasoning. What’s needed is a unified framework that allows the model to take structured, interpretable visual actions—guided by feedback—while remaining grounded in the image content.

Contributions. To address this, we propose a novel framework for inference-time visual token scaling, enabling MLLMs to engage in *iterative, verifier-guided reasoning* over images. Our contributions are threefold:

- **Expressive and theoretically grounded framework.** We formulate visual reasoning as a Markov Decision Process (MDP) with two key components: a *reasoner* that proposes visual actions, and a *verifier*, trained via multi-step Direct Preference Optimization (DPO), which evaluates action quality and terminates reasoning when appropriate. We prove that our reasoner and verifier cooperation system ensures alignment between reasoning actions and visual content, while guaranteeing a bounded number of steps through an early stopping mechanism.
- **A new dataset for tool-augmented visual reasoning.** We introduce a two-part dataset: *VTS-SFT* (supervised reasoning trajectories with tool use), and *VTS-DPO* (preference-labeled reasoning pairs). These resources enable effective training of both the reasoner and verifier, supporting dynamic visual interaction with multi-step grounding.
- **Comprehensive evaluation and state-of-the-art results.** Our experiments span a variety of vision-language tasks demanding multi-step reasoning. Across these scenarios, our approach significantly outperforms strong baselines, including models augmented with limited tool use and those employing chain-of-thought prompting. We observe not only accuracy improvements but also more interpretable reasoning traces, as the model’s step-by-step process explicitly justifies each answer with visual evidence. These results underscore the effectiveness of inference-time visual token scaling: by enabling an AI to look deeper into images in a controlled, stepwise fashion, we achieve new state-of-the-art performance on tasks that previously stymied conventional MLLMs.

These contributions move us toward flexible, grounded, and interpretable visual reasoning in MLLMs—crucial for building AI systems that can “think with their eyes.”

2 Related Work

Visual reasoning. Visual reasoning in VLMs focuses on integrating visual and textual inputs to enable effective decision-making. Early approaches, such as Shikra [2], applied Chain of Thought (CoT)[26] techniques to visual tasks. Meanwhile, methods like SoM[29] and Scaffolding [13] improved reasoning by leveraging visual anchors, such as segmentation maps. V* [27] introduced a two-step CoT approach for high-resolution visual search, demonstrating the potential of structured reasoning in visual contexts. More recently, efforts have been made to develop improved reasoning datasets for training visual language models (VLMs) [21, 7]. However, most of this work primarily focuses on scaling text tokens during the visual reasoning phase. Limited attention has been given to frameworks that scale visual tokens to address complex visual reasoning tasks.

Visual programming and tool-using. Recent research has focused on integrating visual tools with large language models (LLMs) and vision-language models (VLMs) to address complex visual tasks. These studies aim to leverage LLMs and VLMs to generate code that utilizes external vision tools for solving complex vision problems [11, 8, 22, 31]. For instance, Visprog [8] and ViperGPT [22] prompt VLMs to produce single-step Python code that interacts with external vision tools, while Visual Sketchpad [10] enhances these methods by introducing multi-step reasoning capabilities, enabling VLMs to rethink and correct execution errors. However, these approaches fail to address the critical aspect of iterative visual reasoning, as their reasoning steps are typically limited to only 1–2 steps. This prevents them from performing reflective refinements akin to VLMs, hindering their ability to tackle complex visual tasks that require deeper inference and multi-step adjustments.

Verifier design. Recent advancements have leveraged verifiers to enhance language model reasoning and solution quality. Approaches include deriving reward signals for reasoning [15, 16], combining solution- and step-level verifiers for math problems [33], and using graph search or Monte Carlo rollouts for rationale generation [19, 24]. Training methods range from human annotations for RL with feedback [34] to synthetic data for RL with AI feedback [30]. Some treat verifiers as generative models, scoring solutions via control tokens [12] or likelihoods [17]. Closely related is V-STaR [9], which uses Direct Preference Optimization (DPO) for solution ranking. However, existing verifiers are typically designed for specific tasks and tailored training datasets. Our work utilizes multi-step DPO as a verification mechanism, supported by theoretical guarantees.

3 Visual Tokens Scaling with Verification

We first formally formulate the visual reasoning tasks with visual token scaling and verification.

3.1 Problem Formulation

At the first step, a question-image pair s_1 ⁴ is sampled from some distribution $\mathcal{D}$ as the initial state. For each step $h \geq 2$, we conduct:

- **Planning:** the VLM observes the current state s_h , which is the history of the previous $h - 1$ reasoning steps, i.e., $s_h = (s_{h-1}, t_{h-1}, a_{h-1}, o_{h-1})$, and generates planning of h -th step t_h by some distribution $p(\cdot | s_h)$. t_h can be regarded as the text tokens and determines how to manipulate the image at the h -th step.
- **Action:** based on planning t_h , the VLM further chooses an action $a_h \in \mathcal{A}$ according to policy $\pi(\cdot | t_h)$, where $\mathcal{A}$ is the finite action (module) set. This means that the VLM decides which visual module to implement for a specific instruction t_h .
- **Observation:** in response to the action, the environment then returns a visual observation $o_h = f_{a_h}(t_h)$ for some deterministic function f . Here the visual observation is assumed to be deterministic since it is the code execution result by some visual module, the depth map of an image, for example.

Then we transit to a new state s_{h+1} and a new reasoning step begins.

- **Verification:** after each set of planning, action and observation, a verifier r^* is designed to decide whether to continue this iterative reasoning, or to stop and give out the final result.

We regard the final planning as the final result of the initial question-image pair. Eventually, we will collect a reasoning sequence

$$\tau = (s_1, t_1, a_1, o_1, \dots, t_{H_\tau}, a_{H_\tau}, o_{H_\tau}, t_{H_\tau+1}) = s_{H_\tau+1} \cup \{t_{H_\tau+1}\}, \quad (1)$$

where H_τ (a random variable) is the length of the total reasoning steps for path τ .

3.2 Visual Reasoning with Visual Token Scaling and Verification

To enhance performance in visual reasoning tasks, the reasoning process will consist of two main components: a reasoner capable of generating visual reasoning steps, and a verifier that guides the reasoner in visual token scaling and determines the terminal condition.

Reasoner. The reasoner is a pre-trained VLM augmented with plug-and-play modules, which are denoted by R_{θ_0} , where θ_0 encapsulates all the parameters. The module tools can be off-the-shelf vision models (depth maps, bounding box generation, etc), table operations (add columns, select row, etc), web search engines, and Python functions (plot figures, calculation, etc). The reasoner is supposed to present the reasoning steps in a sequential format as in equation (1), where $p(\cdot | s_h) = R_{\theta_0}(\cdot | s_h)$ and $\pi(\cdot | t_h) = R_{\theta_0}(\cdot | t_h)$. In practice, the reasoner can be: (i) a model such as GPT-4o; or (ii) an open-source model fine-tuned on self-crafted datasets. In this case, θ_0 can be further updated to $\hat{\theta}_{\text{SFT}}$ by SFT.

Verifier. The verifier r^* is a function that maps the reasoning trajectory generated by the reasoner to a real number. At each reasoning step h , the reasoner will continue to generate one additional step if the reward difference between step h and step $h + 1$ is no less than some predetermined positive threshold ϵ . In this paper, we utilize a verifier derived from multi-step DPO [28]. Specifically, we begin with a base VLM V_{ϕ_0} , capable of generating reasoning sequences, where ϕ_0 is known and encompasses all the parameters. Using multi-step DPO with preference data, ϕ_0 is further updated to $\hat{\phi}_{\text{SDPO}}$. Consequently, the verifier can be represented by V_{ϕ_0} and $V_{\hat{\phi}_{\text{SDPO}}}$ jointly.

3.3 Reasoner and Verifier Training

The training stage is divided into two parts: fine-tuning the reasoner with SFT and verifier training based on DPO (answer level and multi-step level). Correspondingly, the training dataset consists $\mathcal{D}_{\text{SFT}}$ and $\mathcal{D}_{\text{DPO}}$, we will describe the details of Visual Tokens Scaling SFT and DPO training dataset construction in Section 4.

Fine-tune the reasoner by SFT. Suppose we have a base reasoning model R_{θ_0} .

SFT training. Given a Visual Token Scaling SFT training dataset $\mathcal{D}_{\text{SFT}}$, the supervised finetuning process refers to the learning of reasoner R_θ through minimizing the following cross-entropy loss

$$\hat{\theta}_{\text{SFT}} := \operatorname{argmin}_{\theta} \frac{1}{|\mathcal{D}_{\text{SFT}}|} \sum_{\tau \in \mathcal{D}_{\text{SFT}}} \frac{1}{H_\tau + 1} \mathcal{L}_{\text{SFT}}(R_\theta(\tau), \tau),$$

⁴In practice, s_1 will also include system prompts.

where $\mathcal{L}_{\text{SFT}}(\cdot, \cdot)$ is defined as

$$\mathcal{L}_{\text{SFT}}(\mathbf{R}_\theta(\tau), \tau) = - \sum_{h=1}^{H_\tau+1} \log \mathbf{R}_\theta(t_h^\tau | s_h^\tau) - \sum_{h=1}^{H_\tau} \log \mathbf{R}_\theta(a_h^\tau | t_h^\tau). \quad (2)$$

$\hat{\theta}_{\text{SFT}}$ then can be obtained by the iteration of gradient descent

$$\theta_{k+1} = \theta_k - \alpha \frac{1}{|\mathcal{D}_{\text{SFT}}|} \sum_{\tau \in \mathcal{D}_{\text{SFT}}} \frac{1}{H_\tau + 1} \nabla \mathcal{L}_{\text{SFT}}(\mathbf{R}_\theta(\tau), \tau), \quad (3)$$

where α is the learning rate and the initial parameter is θ_0 .

Verifier training by DPO. We use multi-step DPO to derive the desired verifier that aligns with human preferences, building upon a base VLM V_{ϕ_0} , such as LLaVA-v1.5-7B. Suppose we are given preference pairs (s_1, τ^w, τ^l) , where τ^w is preferred trajectory over unpreferred trajectory τ^l .

The key idea of multi-step DPO is to assume that r^* belongs to a family of one-parameter functions $\{r_\phi(\cdot)\}_\phi$, i.e., $r^*(\cdot) = r_{\phi_{\text{SDPO}}}(\cdot)$ for some unknown ground truth parameter ϕ_{SDPO} . Specifically, if $r_\phi(\cdot)$ is defined as

$$r_\phi(\tau) = \eta \sum_{h=1}^{H_\tau+1} \log \frac{V_\phi(t_h | s_h)}{V_{\phi_0}(t_h | s_h)} + \eta \sum_{h=1}^{H_\tau} \log \frac{V_\phi(a_h | t_h)}{V_{\phi_0}(a_h | t_h)} + Q(s_1),$$

where η is a positive constant and $Q(\cdot)$ is a fixed function that depends only on s_1 , then the preference model $V_{\phi_{\text{SDPO}}}$ will align with the verifier $r_{\phi_{\text{SDPO}}}$ while remaining close to the original V_{ϕ_0} automatically, as shown by Equation 8 and Proposition A.1.

Furthermore, ϕ_{SDPO} can be determined by maximizing the expected likelihood under the Bradley-Terry model

$$\mathbb{E}_{(s_1, \tau^w, \tau^l)} [\mathbb{P}(\tau^w \succ \tau^l)], \quad (4)$$

where

$$\mathbb{P}(\tau^w \succ \tau^l) = \sigma(r^*(\tau^w) - r^*(\tau^l)). \quad (5)$$

Once ϕ_{SDPO} is obtained, the verifier $r_{\phi_{\text{SDPO}}}$ can be used to guide the reasoner in generating the reasoning procedure. The detailed training procedure is as follows.

Multi-step DPO. Define the empirical multi-step DPO loss $\mathcal{L}_{\text{SDPO}}(\phi, \phi_0)$ by equation (11). Let

$$\hat{\phi}_{\text{SDPO}} = \text{argmin}_\phi \mathcal{L}_{\text{SDPO}}(\phi, \phi_0),$$

then $\hat{\phi}_{\text{SDPO}}$ can be obtained by the iteration of gradient descent

$$\phi_{k+1} = \phi_k - \alpha' \sum_{(s_1, \tau^w, \tau^l) \in \mathcal{D}_{\text{DPO}}} \nabla \mathcal{L}_{\text{SDPO}}(\phi, \phi_0), \quad (6)$$

where α' is the learning rate and the initial parameter is ϕ_0 . The verifier r^* operates on some reasoning trajectory τ could be approximated by $r_{\hat{\phi}_{\text{SDPO}}}(\tau)$.

3.4 Inference Algorithm with Practical Efficiency and Theoretical Guarantees

In this subsection, we show our algorithm during inference time and the theoretical guarantees.

Given a new test pair $s_1 \sim \mathcal{D}$, we use the reasoner $\mathbf{R}_{\hat{\theta}_{\text{SFT}}}$ for scaling the inference sequence, and we use the verifier $V_{\hat{\phi}_{\text{SDPO}}}$ to instruct reasoner whether to continue generating the reasoning sequence. Assume a reasoning sequence s_h is generated by the reasoner $\mathbf{R}_{\hat{\theta}_{\text{SFT}}}$. We determine whether $\mathbf{R}_{\hat{\theta}_{\text{SFT}}}$ need to generate one additional reasoning step

$$\begin{aligned} t_h &\sim \mathbf{R}_{\hat{\theta}_{\text{SFT}}}(\cdot | s_h), \quad a_h \sim \mathbf{R}_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h), \quad o_h = f_{a_h}(t_h), \\ s_{h+1} &= (s_h, t_h, a_h, o_h), \end{aligned} \quad (7)$$

by checking whether $|r_{\hat{\phi}_{\text{SDPO}}}(s_{h+1}) - r_{\hat{\phi}_{\text{SDPO}}}(s_h)| \geq \epsilon$, where ϵ is some predetermined positive threshold. We repeat such procedure unless $|r_{\hat{\phi}_{\text{SDPO}}}(s_{h+1}) - r_{\hat{\phi}_{\text{SDPO}}}(s_h)| < \epsilon$. The following lemma gives an equivalent explicit termination condition.

Lemma 3.1. *The reasoner R_{θ_0} stops at the reasoning step h if*

$$\left| r_{\hat{\phi}_{SDPO}}(s_{h+1}) - r_{\hat{\phi}_{SDPO}}(s_h) \right| < \epsilon \Leftrightarrow \left| \log \frac{V_{\hat{\phi}_{SDPO}}(t_h | s_h)}{V_{\phi_0}(t_h | s_h)} + \log \frac{V_{\hat{\phi}_{SDPO}}(a_h | t_h)}{V_{\phi_0}(a_h | t_h)} \right| < \epsilon.$$

Algorithm 1 characterizes the full training and test procedure of our method.

Given $s_1 \sim \mathcal{D}$, denote by $H_\tau(\hat{\phi}_{SDPO}, \phi_0; \hat{\theta}_{SFT})$ ⁵ the total reasoning step given by Algorithm 1, the next theorem shows that $H_\tau(\hat{\phi}_{SDPO}, \phi_0; \hat{\theta}_{SFT})$ is finite almost surely (a.s.).

Theorem 3.2 (Reasoning steps characterization). *The total reasoning step satisfies that*

1. $H_\tau(\hat{\phi}_{SDPO}, \phi_0; \hat{\theta}_{SFT})$ is a stopping time.
2. (Informal). Under some mild condition, $H_\tau(\hat{\phi}_{SDPO}, \phi_0; \hat{\theta}_{SFT})$ is finite with probability 1.

Theorem 3.2 shows that our algorithm is not only flexible—allowing variational reasoning steps that improve upon the results of Xiong et al. [28]—but also guarantees a bounded number of reasoning steps. This ensures that our method avoids scenarios in which the reasoner would otherwise produce looping or repetitive outputs. The detailed proof can be found in Appendix A.4. The superiority of our algorithm can be shown in the following experiments.

4 Visual Tokens Scaling Dataset Construction

As discussed in Section 3, our goal is to train a *reasoner* capable of performing visual token scaling—namely, learning to invoke visual understanding tools throughout the reasoning process to generate intermediate images rich in detailed visual token information. At the same time, we aim to train a *verifier* that can distinguish between two reasoning trajectories of similar structure for the same task, identifying the one that more effectively scales visual information. This collaborative setup enables the verifier to guide and refine the reasoner’s visual reasoning process.

However, a major challenge lies in the absence of existing datasets that contain long-chain, tool-grounded reasoning traces with rich intermediate visual states. In particular, no existing datasets provide both detailed visual CoT-style supervision and curated accepted–rejected trajectory pairs required for DPO training. To address this, we construct a dedicated dataset by building upon the single-image dataset of the LLaVA-OneVision dataset (LLaVA-OV) [14], which contains 3.2M vision-language examples covering a broad range of tasks.

In this section, we describe our dataset construction pipeline in detail. Section 4.1 introduces how we collect and preprocess visual reasoning examples from LLaVA-OV to produce structured tool-based trajectories. Section 4.2 then explains how we derive both the SFT and DPO datasets from these verified trajectories.

4.1 Visual Token Scaling Dataset Collection and Preprocessing

We begin our dataset construction by sampling image-question pairs from the single-image portion of the LLaVA-OV dataset, which covers a wide range of vision-language tasks such as grounding, chart understanding, OCR, and mathematical reasoning. To ensure sufficient task diversity, we uniformly sample across all 83 task types included in the dataset.

For each sampled example, we employ Qwen-2.5-VL-72B in a few-shot prompting setting to generate a visual token scaling trajectory. This trajectory is composed of a sequence of tool-based operations drawn from our predefined action space $\mathcal{A}$ (as described in Section 3.1). In line with prior work [8, 11], the action space comprises ten visual tools: GroundingAction, DepthAction, ZoomInAction, VisualSearchAction, Crop, OCR, ImageSegment, ImageCaptioner, SimilarityComputing, and Overlay. These tools allow the model to iteratively perceive, transform, and enrich the visual content throughout the multi-step reasoning process.

Generated trajectories may include errors, invalid tool use, or inconsistent reasoning. We apply a preprocessing pipeline to flatten nested structures, merge related turns, remove malformed examples, and ensure metadata consistency. We also filter out trivial cases with no tool use and those exceeding 20,000 tokens due to memory limits. To guarantee the correctness of each trajectory, we employ

⁵For notation simplicity, here we omit the dependence on s_1 .

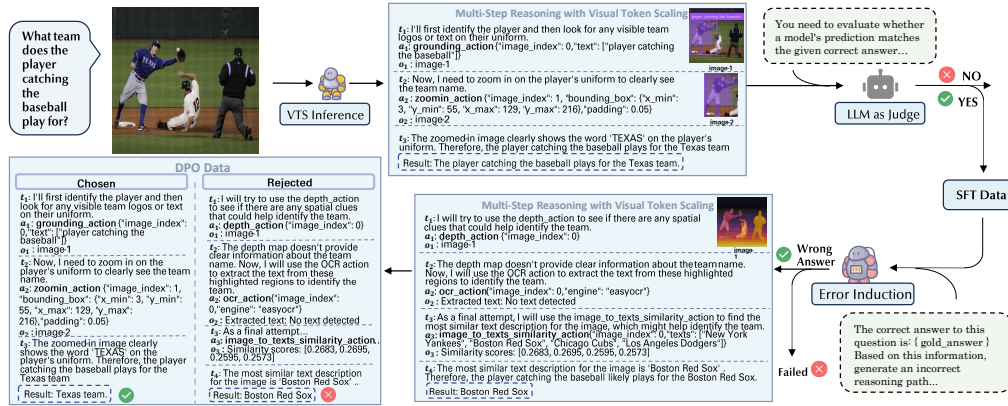


Figure 2: **Pipeline for Synthetic Data Generation and Curation in VTS-V.** Our data construction process consists of three stages: (1) generating multi-step reasoning trajectories with visual tool calls, (2) filtering out incorrect trajectories using an LLM-as-a-judge framework, and (3) creating contrastive (correct vs. incorrect) trajectory pairs for multi-step DPO training.

the same Qwen-2.5-VL-72B model as an LLM-based verifier (`llm_as_a_judge`) to filter out those whose final answers are deemed incorrect. Starting from over 650K generated trajectories, this process results in a curated set of 315K high-quality visual token scaling examples, which serve as the basis for supervised fine-tuning and preference training.

4.2 VTS-SFT and VTS-DPO Dataset Construction

Starting from the 315K verified visual token scaling trajectories described above, we construct two datasets: one for supervised fine-tuning (VTS-SFT) and one for direct preference optimization (VTS-DPO).

VTS-SFT Dataset Construction. To construct the VTS-SFT dataset, we transform each verified trajectory into a supervised training instance. Each trajectory τ follows the reasoning path format defined in Equation 1, consisting of a sequence of textual states, tool actions, and visual observations, concluding with a final answer. We retain trajectories where the final answer satisfies $t_{H_\tau+1} = t^*$, where t^* is the ground-truth label associated with the input s_1 . The resulting supervised dataset is defined as:

$$\mathcal{D}_{\text{SFT}} = \{(s_1, \tau) \mid t_{H_\tau+1} = t^*, \text{llm_as_a_judge}(\tau) = \text{correct}\}.$$

Each trajectory in $\mathcal{D}_{\text{SFT}}$ is tool-grounded and preserves all intermediate reasoning states, providing rich supervision for training. After processing, the dataset contains approximately 315K high-quality examples.

VTS-DPO Dataset Construction. To construct the preference dataset $\mathcal{D}_{\text{DPO}}$, we generate suboptimal reasoning trajectories as contrastive pairs. For each $(s_1, \tau^w) \in \mathcal{D}_{\text{SFT}}$, where τ^w is a correct trajectory, we design prompts that instruct the model to begin from an intermediate point and then proceed with incorrect reasoning steps, yielding a wrong final answer, which is also the Error Induction step in Figure 4. The resulting suboptimal trajectory τ^l is paired with τ^w to form a preference tuple (s_1, τ^w, τ^l) .

We format these pairs into the standard DPO training structure and remove any example where either τ^w or τ^l contains empty or missing image inputs. After filtering, the final dataset $\mathcal{D}_{\text{DPO}}$ comprises 301K preference pairs for training the verifier to assess and guide visual token scaling quality.

5 Experiments

We conduct comprehensive experiments to evaluate the effectiveness of our proposed Visual Token Scaling with Verification (VTS-V) framework across a range of visual reasoning tasks. This section is organized as follows: In Section 5.1, we describe the experimental setup, including model variants, training configurations, and evaluation benchmarks. Section 5.2 presents our end-to-end experiments, comparing VTS-V to several strong baselines in both closed-source and open-source settings. Finally, in Section 5.3, we conduct ablation studies to analyze the individual contributions of visual token scaling and verifier integration.

Model	Depth	Spatial	Jigsaw	VisCorr	SemCorr	ArtStyle	Count	FunCorr	Local	MultiV	Refl	Fore	Sim	Avg
Qwen2.5-VL-7B-Instruct Variants														
Qwen2.5VL-7B	65.32	83.22	56.67	40.12	24.46	58.97	65.00	19.23	41.80	43.61	25.37	34.85	79.26	49.07
Qwen2.5VL-7B + VTS-V	66.13	37.76	53.33	58.72	36.69	58.97	41.18	23.85	45.08	42.86	26.12	38.64	64.44	45.67
Ours + VTS-V	70.97	86.01	68.67	54.44	33.81	67.52	65.83	30.00	49.18	55.64	38.06	36.36	80.00	56.65
Qwen2-VL-7B-Instruct Variants														
Qwen2VL-7B	57.26	79.72	54.00	33.72	31.65	51.28	73.33	18.46	54.10	45.11	33.58	38.64	53.33	48.01
Qwen2VL-7B + VTS-V	49.19	67.83	57.05	15.72	17.56	42.74	43.33	12.90	49.59	30.83	32.84	29.55	51.85	38.54
Ours + VTS-V	60.48	60.48	58.00	37.21	40.58	76.92	63.33	28.35	51.64	47.37	35.82	34.09	84.21	52.19
LLaMA-3.2-11B-Vision-Instruct Variants														
LLaMA3.2-11B	63.71	67.13	53.33	50.58	39.57	47.86	55.00	32.31	62.30	48.12	31.34	25.76	46.67	47.98
LLaMA3.2-11B + VTS-V	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Ours + VTS-V	68.55	69.23	57.33	38.60	47.48	55.56	56.67	35.43	58.20	52.63	33.58	23.48	57.46	50.32
GPT-4o Variants														
GPT-4o	74.19	69.23	55.33	75.00	53.96	82.91	49.17	40.77	59.84	59.40	37.31	79.55	72.59	62.25
GPT-4o + Sketchpad	83.90	81.10	70.70	80.80	58.30	77.19	66.70	42.10	65.40	45.60	33.10	79.00	84.20	66.78
GPT-4o + CoT	73.39	82.52	62.00	82.56	57.55	82.05	65.00	57.69	60.66	53.38	41.04	62.88	63.70	64.96
GPT-4o + SoM	68.55	76.22	49.33	83.72	52.52	-	43.33	47.69	59.84	56.40	-	-	63.70	60.13
GPT-4o + MMFactory	80.30	81.80	75.30	85.50	58.30	83.00	61.70	55.40	59.00	60.20	35.10	84.80	75.30	68.90
GPT-4o + VTS-V (Ours)	79.84	85.31	75.33	82.56	56.83	80.34	67.50	53.08	68.85	52.63	40.30	71.21	85.19	69.15

Table 1: **Model performance on BLINK subtasks.** Each column corresponds to a different visual reasoning task in the BLINK benchmark. Highlighted cells show whether using the proposed VTS-V method improves or degrades the performance compared to the base model.

5.1 Experimental Setup

Models and Baselines. Our experiments are divided into two main parts: closed-source models and open-source models. For the closed-source setting, we follow prior work and evaluate the effectiveness of VTS-V using GPT-4o, comparing it with four strong baselines: (i) Zero-shot reasoning on GPT-4o; (ii) Chain-of-Thought (CoT) reasoning on GPT-4o; (iii) Visual prompting framework: Set-of-Mark (SoM), and Visual Sketchpad; and (iv): Tool-using framework: MMFactory.

For the open-source setting, we fine-tune three vision-language model: Qwen2-VL-7B-Instruct [25], Qwen2.5-VL-7B-Instruct[1], and LLaMA3.2-Vision-Instruct-11B on our VTS-SFT dataset. We evaluate the performance of VTS-V both before and after fine-tuning to assess its impact. Our verifier model is trained on Qwen2.5-VL-7B-Instruct using our VTS-DPO dataset. All SFT experiments are conducted using LLaMA Factory under unified settings, and DPO training is carried out with TRL. Detailed training configurations are provided in Appendix B.

Evaluation Tasks. We evaluate on four representative vision-language reasoning benchmarks: BLINK [6], V*Bench [27], MMStar [3], and MathVista [18]. These benchmarks span a diverse set of capabilities, including multi-image perception and reasoning (BLINK), fine-grained understanding of small visual objects (VBench), broad general knowledge QA (MMStar), and visual mathematical reasoning (MathVista). Following [11] and [5], we conduct our main experiments on 13 tasks from BLINK, a challenging and diverse benchmark designed to evaluate fine-grained visual reasoning. These tasks span visual and semantic correspondence, spatial understanding, multi-view reasoning, depth and reflectance estimation, counting, object localization, pattern alignment (e.g., jigsaw), art style comparison, functional and semantic matching, visual similarity, and forensic image detection. Further evaluation details are provided in the Appendix.

5.2 End-to-End Experiments

VTS-V improves performance of closed-source models. As shown in Table 1, our VTS-V framework brings robust and consistent improvements to GPT-4o on the BLINK benchmark. The significant performance gain, achieved without fine-tuning, demonstrates the inherent effectiveness of our verifier-guided visual token scaling method for enhancing visual reasoning. Specifically, the average performance increases from 62.25% to 69.15% (+6.90), outperforming all other prompting-based and tool-augmented variants. Compared with GPT-4o + MMFactory, VTS-V yields a slight but meaningful gain of +0.25. It shows larger improvements over GPT-4o + CoT (+3.30), GPT-4o + SoM (+9.02), and GPT-4o + Sketchpad (+2.37). The largest gains are observed on complex compositional tasks, such as Counting (+18.33), Functionally-Correlated (+12.91), and Visually-Correlated (+7.56), indicating that VTS-V strengthens reasoning over fine-grained and structured visual information.

VTS-V enhances open-source models fine-tuned on our dataset. Open-source models also benefit significantly from our framework. When these models are first fine-tuned on our VTS-SFT dataset and then evaluated using VTS-V, consistent gains are observed. For example, Qwen2VL-7B improves from 48.01% to 52.19% (+4.18), showing gains in Multi-view (+3.44), Local (+2.48), and

Jigsaw (+4.08). Similarly, Qwen2.5VL-7B improves from 49.07% to 56.65% (+7.58), with boosts in Depth (+5.65), Spatial (+2.79), and Visual Correlation (+14.32). LLaMA3.2-11B, despite being a larger model, shows an improvement from 47.98% to 50.32% (+2.34), especially on tasks like Semantic Correlation (+8.89) and Functional Correlation (+3.12). These results demonstrate that VTS-V generalizes well to open-source architectures and consistently improves performance across vision reasoning categories.

Fine-tuning on VTS-SFT datasets enhances models’ tool-using abilities. We find that applying VTS-V directly to open-source models without supervised fine-tuning can harm performance. For instance, applying VTS-V to Qwen2VL-7B drops performance from 48.01% to 38.54%, and LLaMA3.2-11B fails to output any valid tool-use actions. This behavior contrasts with GPT-4o, which can reliably follow verifier guidance even without additional training. These results suggest that base open-source models do not possess the necessary behavior patterns to interact properly with verifier signals or external tools. However, after supervised fine-tuning on our VTS-SFT dataset, these models not only recover but surpass their original performance. Post-finetuning, Qwen2VL-7B improves to 52.19%, Qwen2.5VL-7B rises to 56.65%, and LLaMA3.2-11B reaches 50.32%. More importantly, they now produce valid tool-use actions, indicating successful alignment with verifier guidance. This confirms that the VTS-SFT dataset is critical for enabling reliable tool interaction in open-source VLMs.

VTS-V generalizes to diverse benchmarks. To assess generalization, we evaluate models on V*Bench[27], MMStar[3], and MathVista[18] (Table 2). Without fine-tuning, applying VTS-V can degrade performance—for instance, Qwen2.5VL-7B drops from 73.30 to 67.54 on VBench. After full training (VTS-SFT + VTS-V), performance improves across all models: Qwen2.5VL-7B reaches 75.13, Qwen2VL-7B recovers to 66.67, and LLaMA3.2-11B rises to 60.21. The results for GPT-4o on these benchmarks can be found in Table 5.

5.3 Ablation Study and Discussion

We compare GPT-4o + VTS-V with a text-only chain-of-thought baseline (GPT-4o + CoT). VTS-V achieves 69.15% average accuracy, surpassing CoT’s 64.96% by +4.19 points. The gains are especially strong on tasks needing structured visual reasoning, such as Counting (+10.90), Functionally-Correlated (+10.98), and Visually-Correlated (+1.76). These results highlight that scaling visual reasoning via verifier-guided tool use is more effective than relying solely on extended textual reasoning.

Verifier improves reasoning quality.

Ablation results in Table 3 clearly demonstrate the effectiveness of the verifier module. When the verifier is removed (“w/o verifier”), accuracy consistently drops across all models—Qwen2.5VL sees a decline from 54.44 to 47.09, Qwen2VL from 37.21 to 33.14, and Llama3.2vision from 38.60 to 36.63. This indicates that the verifier plays a crucial role in improving reasoning by filtering out suboptimal answer paths and enhancing decision quality. These improvements are even more pronounced compared to removing the whole VTS and verifier, which leads to further substantial accuracy drops.

6 Conclusions and Limitations

This paper presents a new framework, Visual Token Scaling with Verification (VTS-V), to enhance multi-step visual reasoning by iteratively selecting visual actions and verifying their utility at each

Trained Model	V*Bench	MMStar	MathVista
Qwen2.5VL-7B	73.30	55.00	67.50
Qwen2.5VL-7B + VTS-V	67.54	55.93	52.60
Qwen2.5VL-7B (Ours) + VTS-V	75.13	57.93	66.50
Qwen2VL-7B	66.49	53.93	58.20
Qwen2VL-7B + VTS-V	50.80	44.29	37.50
Qwen2VL-7B (Ours) + VTS-V	66.67	55.26	60.30
LLaMA-3.2-11B	54.45	50.57	32.20
LLaMA-3.2-11B + VTS-V	—	—	—
LLaMA-3.2-11B (Ours) + VTS-V	60.21	50.71	48.10

Table 2: **Model performance on general benchmarks.** Evaluation on V*Bench, MMStar, and MathVista. Dashes indicate missing results.

	Qwen2.5VL	Qwen2VL	Llama3.2vision
Full	54.44	37.21	38.60
w/o verifier	47.09	33.14	36.63
w/o VTS-V	40.70	19.77	31.98

Table 3: Ablation results: accuracy drops when removing verifier or VTS-V.

step. By framing the process as a Markov Decision Process and integrating a verifier trained via step-wise Direct Preference Optimization, the approach enables models to progressively refine their understanding of complex visual inputs. VTS-V achieves strong performance across a range of challenging visual reasoning benchmarks, outperforming existing baselines.

Limitations. The framework can be adapted to support a broader set of visual tools and more diverse reasoning formats. Enhancing the verifier’s adaptability across tasks and integrating it more tightly with downstream applications may further improve robustness. We believe VTS-V offers a flexible foundation for future research in compositional and tool-augmented visual reasoning.

References

- [1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- [2] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal llm’s referential dialogue magic. *arXiv preprint arXiv:2306.15195*, 2023.
- [3] 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? *arXiv preprint arXiv:2403.20330*, 2024.
- [4] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.
- [5] Wan-Cyuan Fan, Tanzila Rahman, and Leonid Sigal. Mmfactory: A universal solution search engine for vision-language tasks. *arXiv preprint arXiv:2412.18072*, 2024.
- [6] 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 *European Conference on Computer Vision*, pp. 148–166. Springer, 2025.
- [7] Jarvis Guo, Tuney Zheng, Yuelin Bai, Bo Li, Yubo Wang, King Zhu, Yizhi Li, Graham Neubig, Wenhu Chen, and Xiang Yue. Mammoth-vl: Eliciting multimodal reasoning with instruction tuning at scale. *arXiv preprint arXiv:2412.05237*, 2024.
- [8] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14953–14962, 2023.
- [9] Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.
- [10] Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. *Advances in Neural Information Processing Systems*, 37:139348–139379, 2024.
- [11] Yushi Hu, Otilia Stretcu, Chun-Ta Lu, Krishnamurthy Viswanathan, Kenji Hata, Enming Luo, Ranjay Krishna, and Ariel Fuxman. Visual program distillation: Distilling tools and programmatic reasoning into vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9590–9601, 2024.
- [12] Tomasz Korbak, Kejian Shi, Angelica Chen, Rasika Vinayak Bhalerao, Christopher Buckley, Jason Phang, Samuel R Bowman, and Ethan Perez. Pretraining language models with human preferences. In *International Conference on Machine Learning*, pp. 17506–17533. PMLR, 2023.
- [13] Xuanyu Lei, Zonghan Yang, Xinrui Chen, Peng Li, and Yang Liu. Scaffolding coordinates to promote vision-language coordination in large multi-modal models. *arXiv preprint arXiv:2402.12058*, 2024.
- [14] 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.
- [15] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5315–5333, 2023.
- [16] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.

- [17] Yixin Liu, Avi Singh, C Daniel Freeman, John D Co-Reyes, and Peter J Liu. Improving large language model fine-tuning for solving math problems. *arXiv preprint arXiv:2310.10047*, 2023.
- [18] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.
- [19] Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. Let’s reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*, 2023.
- [20] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.
- [21] Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Unleashing chain-of-thought reasoning in multi-modal language models. *arXiv preprint arXiv:2403.16999*, 2024.
- [22] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11888–11898, 2023.
- [23] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9568–9578, 2024.
- [24] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*, 2023.
- [25] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.
- [26] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [27] Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13084–13094, 2024.
- [28] Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, et al. Building math agents with multi-turn iterative preference learning. *arXiv preprint arXiv:2409.02392*, 2024.
- [29] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023.
- [30] Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. Rlcd: Reinforcement learning from contrastive distillation for lm alignment. In *The Twelfth International Conference on Learning Representations*.
- [31] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.
- [32] Tong Zhang. *Mathematical analysis of machine learning algorithms*. Cambridge University Press, 2023.
- [33] Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujia Yang. Solving math word problems via cooperative reasoning induced language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4471–4485, 2023.
- [34] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

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: Our motivation, contribution, and scope are clearly described in both the abstract and the introduction.

Guidelines:

- The answer NA 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 NA 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: Justification: We describe our limitations in Sec. 6.

Guidelines:

- The answer NA 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: [\[Yes\]](#)

Justification: Our proof is in Appendix. [A](#).

Guidelines:

- The answer NA 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 includes all necessary details for reproducing the main experimental results in Section [5.1](#) and [B](#). This includes a thorough description of the model architecture, training procedures (e.g., optimizer settings, batch size, learning rate schedules, number of training epochs), dataset splits, and evaluation metrics in . We also provide hyperparameter settings and implementation details that influence performance. To further support reproducibility, we will release the source code and datasets for training and evaluation upon publication.

Guidelines:

- The answer NA 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: The code and data are in supplementary materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.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://nips.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, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We present our experimental setting and details in Appendix B.

Guidelines:

- The answer NA 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: Since our work focus on inference of language models, the results are provided as evaluation scores on validation/test set. And we have explained how they are calculated in Section 5.1.

Guidelines:

- The answer NA 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: We show the experiments compute resources in B.

Guidelines:

- The answer NA 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 strictly followed it in this work.

Guidelines:

- The answer NA 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:

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper poses no such risks.

Guidelines:

- The answer NA 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: We use one public datasets and strictly follow their usage license. Besides, we cite their original paper to express our gratitude for their work.

Guidelines:

- The answer NA 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: This paper does not release new assets.

Guidelines: We includes the anonymized URL as well as details about dataset and code in supplementary material.

- The answer NA 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: [NA]

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

Guidelines:

- The answer NA 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: [NA]

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

Guidelines:

- The answer NA 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: [\[Yes\]](#)

Justification: We utilize LLMs as one of the key components for generating the instruction data, and we have provided **ALL** the prompts used in our experiments.

Guidelines:

- The answer NA 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 (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

Appendices

A	Proofs	20
A.1	Verifier Obtaining	20
A.2	Proof of Lemma 3.1	22
A.3	Practical Algorithm for VTS	22
A.4	Proof of Theorem 3.2	22
B	Experimental Details	24
C	Examples	25
C.1	VTS-DPO Dataset Examples	25
C.2	VTS-SFT Dataset Examples	28
C.3	Benchmark Examples	31
D	Broader Impact	33
E	Inference-Time Overhead Analysis	33
F	Extended Evaluation on Broader Benchmarks	35
F.1	Experimental Setup	35
F.2	Results and Analysis	35
G	Analysis of Performance Differences Between GPT-4o and Open-Source Models	35
H	Visual Token Scaling Analysis with Optimal-Stopping Verifier	36
I	Robustness and Generalization Analysis of the Verifier	36
J	Separate Reasoner and Verifier	37
K	Interpretability of Reasoning Traces	37
L	Ablation Study on the Reasoner Module	38
M	Comparison with Dynamic Visual Attention Methods	38
N	Novelty Analysis and Comparison with Prior Work	39

A Proofs

In this section, we provide the detailed proof that is omitted in the main paper.

A.1 Verifier Obtaining

Given the base model V_{ϕ_0} and preference pairs (s_1, τ^w, τ^l) , the key idea of multi-step DPO (Rafailov et al. [20]; Xiong et al. [28]) is to assume that r^* has a specific structure under which the new preference model $V_{\phi_{\text{SDPO}}}$ and verifier r^* can be obtained by jointly solving the following KL-regularized planning problem and the maximum likelihood estimation of the Bradley-Terry model.

$$\phi_{\text{SDPO}} \in \operatorname{argmin}_{\phi} \mathbb{E}_{H_{\tau}} \mathbb{E}_{s_1 \sim \mathcal{D}, \{t_h \sim V_{\phi}(\cdot | s_h)\}_{h=1}^{H_{\tau}+1}, \{a_h \sim V_{\phi}(\cdot | t_h), o_h = f_{a_h}(t_h)\}_{h=1}^{H_{\tau}}} \left[-r^*(\tau) + \eta \sum_{h=1}^{H_{\tau}+1} D_{\text{KL}}(V_{\phi}(\cdot | s_h) || V_{\phi_0}(\cdot | s_h)) - \eta \sum_{h=1}^{H_{\tau}} D_{\text{KL}}(V_{\phi}(\cdot | t_h) || V_{\phi_0}(\cdot | t_h)) \right], \quad (8)$$

$$\phi_{\text{SDPO}} \in \operatorname{argmin}_{\phi} -\mathbb{E}_{(s_1, \tau^w, \tau^l)} [\mathbb{P}(\tau^w \succ \tau^l)] = \operatorname{argmin}_{\phi} -\mathbb{E}_{(s_1, \tau^w, \tau^l)} [\sigma(r^*(\tau^w) - r^*(\tau^l))]. \quad (9)$$

Firstly, the next proposition shows that problem (8) can be solved directly when r^* lies in a specific family of one-parameter functions,.

Proposition A.1. *If $r^*(\tau) = r_{\phi'}(\tau) = \eta \sum_{h=1}^{H_{\tau}+1} \log \frac{V_{\phi'}(t_h | s_h)}{V_{\phi_0}(t_h | s_h)} + \eta \sum_{h=1}^{H_{\tau}} \log \frac{V_{\phi'}(a_h | t_h)}{V_{\phi_0}(a_h | t_h)} + Q(s_1)$ for some ϕ' , where $Q(\cdot)$ is some fixed function that only depends on s_1 , then the optimal solution of problem (8) is ϕ' .*

Hence if r^* takes the form $r_{\phi'}$, by Proposition A.1 and equation (8), we have $r^* = r_{\phi_{\text{DPO}}}$. Then to further obtain ϕ_{SDPO} , we can plug $r_{\phi'}(\tau)$ in equation (9) and optimize over ϕ' , i.e.,

$$\begin{aligned} \phi_{\text{SDPO}} \in \operatorname{argmin}_{\phi'} -\mathbb{E}_{(s_1, \tau^w, \tau^l)} [\sigma(r_{\phi'}^*(\tau^w) - r_{\phi'}^*(\tau^l))] \\ = \operatorname{argmin}_{\phi'} -\mathbb{E}_{(s_1, \tau^w, \tau^l)} \left[\log \sigma \left(\eta \left(\sum_{h=1}^{H_{\tau^w}+1} \log \frac{V_{\phi'}(t_h^w | s_h^w)}{V_{\phi_0}(t_h^w | s_h^w)} + \sum_{h=1}^{H_{\tau^w}} \log \frac{V_{\phi'}(a_h^w | t_h^w)}{V_{\phi_0}(a_h^w | t_h^w)} \right. \right. \right. \\ \left. \left. \left. - \sum_{h=1}^{H_{\tau^l}+1} \log \frac{V_{\phi'}(t_h^l | s_h^l)}{V_{\phi_0}(t_h^l | s_h^l)} - \sum_{h=1}^{H_{\tau^l}} \log \frac{V_{\phi'}(a_h^l | t_h^l)}{V_{\phi_0}(a_h^l | t_h^l)} \right) \right) \right]. \end{aligned} \quad (10)$$

Here we use superscript to indicate which trajectory path the reasoning steps lie in. Now we can use the following empirical multi-step DPO loss

$$\begin{aligned} \mathcal{L}_{\text{SDPO}}(\phi, \phi_0) = - \sum_{(s_1, \tau^w, \tau^l) \in \mathcal{D}_{\text{DPO}}} \log \sigma \left(\eta \left(\sum_{h=1}^{H_{\tau^w}+1} \log \frac{V_{\phi'}(t_h^w | s_h^w)}{V_{\phi_0}(t_h^w | s_h^w)} + \sum_{h=1}^{H_{\tau^w}} \log \frac{V_{\phi'}(a_h^w | t_h^w)}{V_{\phi_0}(a_h^w | t_h^w)} \right. \right. \\ \left. \left. - \sum_{h=1}^{H_{\tau^l}+1} \log \frac{V_{\phi'}(t_h^l | s_h^l)}{V_{\phi_0}(t_h^l | s_h^l)} - \sum_{h=1}^{H_{\tau^l}} \log \frac{V_{\phi'}(a_h^l | t_h^l)}{V_{\phi_0}(a_h^l | t_h^l)} \right) \right), \end{aligned} \quad (11)$$

and obtain the optimal $\hat{\phi}_{\text{SDPO}} \in \operatorname{argmin}_{\phi} \mathcal{L}_{\text{SDPO}}(\phi, \phi_0)$ by gradient descent. Finally, we use $r_{\hat{\phi}_{\text{SDPO}}}$ to approximate the verifier $r^* = r_{\phi_{\text{SDPO}}}$.

Proof of Proposition A.1. The key idea is, given any s_1 and H_τ , using backward induction to check the optimality of ϕ' for $1 \leq h \leq H_\tau + 1$. For any given s_h , define the value function $Q_h(s_h, \phi)$ as

$$\begin{aligned} Q_h(s_h, \phi) = & \mathbb{E}_{\{t_{h'} \sim \mathbf{V}_\phi(\cdot | s_{h'})\}_{h'=h}^{H_\tau+1}, \{a_{h'} \sim \mathbf{V}_\phi(\cdot | t_{h'}), o_{h'} = f_{a_{h'}}(t_{h'})\}_{h'=h}^{H_\tau}} \left[-\eta \sum_{h'=h}^{H_\tau+1} \log \frac{\mathbf{V}_{\phi'}(t_{h'} | s_{h'})}{\mathbf{V}_{\phi_0}(t_{h'} | s_{h'})} \right. \\ & + \eta \sum_{h'=h}^{H_\tau+1} D_{\text{KL}}(\mathbf{V}_\phi(\cdot | s_{h'}) || \mathbf{V}_{\phi_0}(\cdot | s_{h'})) \\ & \left. + \eta \mathbf{1}\{h \leq H_\tau\} \sum_{h'=h}^{H_\tau} \left(-\log \frac{\mathbf{V}_{\phi'}(a_{h'} | t_{h'})}{\mathbf{V}_{\phi_0}(a_{h'} | t_{h'})} + D_{\text{KL}}(\mathbf{V}_\phi(\cdot | t_{h'}) || \mathbf{V}_{\phi_0}(\cdot | t_{h'})) \right) \right] \\ & - \mathbf{1}\{h = 1\} Q(s_1). \end{aligned}$$

Then our goal is to show that

$$\phi' \in \operatorname{argmin}_\phi \mathbb{E}_{H_\tau} \mathbb{E}_{s_1 \sim \mathcal{D}} Q_1(s_1, \phi). \quad (12)$$

Observe that by the construction, $Q_h(\cdot, \phi)$ satisfies the following recurrence formula

$$\begin{aligned} Q_h(s_h, \phi) &= \mathbb{E}_{t_h \sim \mathbf{V}_\phi(\cdot | s_h), a_h \sim \mathbf{V}_\phi(\cdot | t_h), o_h = f_{a_{h+1}}(t_h)} Q_{h+1}(s_{h+1}, \phi) - \mathbf{1}\{h = 1\} Q(s_1) \\ &+ \eta \mathbb{E}_{t_h \sim \mathbf{V}_\phi(\cdot | s_h), a_h \sim \mathbf{V}_\phi(\cdot | t_h), o_h = f_{a_{h+1}}(t_h)} \left[-\log \frac{\mathbf{V}_{\phi'}(t_h | s_h)}{\mathbf{V}_{\phi_0}(t_h | s_h)} + D_{\text{KL}}(\mathbf{V}_\phi(\cdot | s_h) || \mathbf{V}_{\phi_0}(\cdot | s_h)) \right] \\ &+ \eta \mathbf{1}\{h \leq H_\tau\} \mathbb{E}_{t_h \sim \mathbf{V}_\phi(\cdot | s_h), a_h \sim \mathbf{V}_\phi(\cdot | t_h), o_h = f_{a_{h+1}}(t_h)} \left[-\log \frac{\mathbf{V}_{\phi'}(a_h | t_h)}{\mathbf{V}_{\phi_0}(a_h | t_h)} + D_{\text{KL}}(\mathbf{V}_\phi(\cdot | t_h) || \mathbf{V}_{\phi_0}(\cdot | t_h)) \right]. \end{aligned} \quad (13)$$

$$(14)$$

We now use backward induction to show that for any fixed s_h , $1 \leq h \leq H_\tau + 1$, $\phi' \in \operatorname{argmin}_\phi Q_h(s_h, \phi)$.

(a). For $h = H_\tau + 1$,

$$Q_{H_\tau+1}(s_{H_\tau+1}, \phi) = \mathbb{E}_{t_{H_\tau+1} \sim \mathbf{V}_\phi(\cdot | s_{H_\tau+1})} \left[-\eta \log \frac{\mathbf{V}_{\phi'}(t_{H_\tau+1} | s_{H_\tau+1})}{\mathbf{V}_{\phi_0}(t_{H_\tau+1} | s_{H_\tau+1})} + \eta D_{\text{KL}}(\mathbf{V}_\phi(\cdot | s_{H_\tau+1}) || \mathbf{V}_{\phi_0}(\cdot | s_{H_\tau+1})) \right].$$

By directly using Lemma A.2, we can conclude that $Q_{H_\tau+1}(s_{H_\tau+1}, \phi) \geq 0$, where the equality holds when $\mathbf{V}_\phi(\cdot | s_{H_\tau+1}) = \mathbf{V}_{\phi'}(\cdot | s_{H_\tau+1})$. Hence $\phi' \in \operatorname{argmin}_\phi Q_{H_\tau+1}(s_{H_\tau+1}, \phi)$.

(b). Assume for $h = h'$, $2 \leq h' \leq H_\tau + 1$, $\phi' \in \operatorname{argmin}_\phi Q_{h'}(s_{h'}, \phi)$. Then for $h = h' - 1$,

$$\phi' \in \operatorname{argmin}_\phi \mathbb{E}_{t_{h'-1} \sim \mathbf{V}_\phi(\cdot | s_{h'-1}), a_{h'-1} \sim \mathbf{V}_\phi(\cdot | t_{h'-1}), o_{h'-1} = f_{a_{h'}}(t_{h'-1})} Q_{h'}(s_{h'}, \phi).$$

To show $\phi' \in \operatorname{argmin}_\phi Q_{h'-1}(s_{h'-1}, \phi)$, by the definition of $Q_{h'-1}(s_{h'-1}, \phi)$, one only needs to show that both term (13) and term (14) obtain the minimum at ϕ' when substituting h to $h' - 1$. This can be checked by using conditional expectation and Lemma A.2.

Hence by combining (a) and (b), we have $\phi' \in \operatorname{argmin}_\phi Q_1(s_1, \phi)$, which implies equation (12), as desired. $\square$

Lemma A.2 (Proposition 7.16 and Theorem 15.3 of Zhang [32]). Let $U(\cdot)$ be a given function and $p_0(\cdot)$ be a given density function. Then the solution of

$$\operatorname{argmin}_{p(\cdot)} \mathbb{E}_{x \sim p(\cdot)} [-U(x) + \eta D_{\text{KL}}(p(\cdot) || p_0(\cdot))]$$

is given by

$$p^*(x) = \frac{1}{C} p_0(x) \exp \left(\frac{U(x)}{\eta} \right),$$

where $C = \log \mathbb{E}_{x \sim p_0(\cdot)} \exp \left(\frac{U(x)}{\eta} \right)$. $p^*(\cdot)$ is known as the Gibbs distribution.

A.2 Proof of Lemma 3.1

Proof of Lemma 3.1. This is directly obtained from the definition of $r_{\hat{\phi}_{\text{SDPO}}}$. □

A.3 Practical Algorithm for VTS

Algorithm 1 Visual Reasoning with Visual Token Scaling and Verification

Require Training dataset $\mathcal{D}_{\text{train}}$, initial reasoner R_{θ_0} and base model V_{ϕ_0} .

Input: A new test question-image pair $s_1 \sim \mathcal{D}$.

Output: Reasoning trajectory τ .

- 1: **Reasoning Sequence Generation:** Initialize reasoning trajectory $\tau \leftarrow \{s_1\}$.
 - 2: Set step counter $h \leftarrow 1$.
 - 3: **while** True **do**
 - 4: **Planning:** Generate step planning $t_h \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | s_h)$.
 - 5: **Action:** Select action $a_h \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h)$.
 - 6: **Observation:** Return $o_h = f_{a_h}(t_h)$.
 - 7: Update State: $s_{h+1} \leftarrow (s_h, t_h, a_h, o_h)$.
 - 8: Append (t_h, a_h, o_h) to reasoning trajectory τ .
 - 9: **Verification:** Compute reward difference $\Delta r = r_{\hat{\phi}_{\text{SDPO}}}(s_{h+1}) - r_{\hat{\phi}_{\text{SDPO}}}(s_h)$.
 - 10: **if** $\Delta r < \epsilon$ **then**
 - 11: Break loop.
 - 12: **end if**
 - 13: $h \leftarrow h + 1$.
 - 14: **end while**
 - 15: Generate the final result $t_{H_\tau(\hat{\phi}_{\text{SDPO}})+1} \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | s_{H_\tau(\hat{\phi}_{\text{SDPO}})+1})$, where $H_\tau(\hat{\phi}_{\text{SDPO}}, \phi_0; \hat{\theta}_{\text{SFT}})$ is the total reasoning step.
 - 16: Return final trajectory $\tau = s_{H_\tau(\hat{\phi}_{\text{SDPO}})+1} \cup \{t_{H_\tau(\hat{\phi}_{\text{SDPO}})+1}\}$.
-

A.4 Proof of Theorem 3.2

Before we proceed with the main proof, we provide some relevant definitions and conclusions at the beginning.

Definition A.3 (Martingale, supermartingale, and submartingale). Let $\mathcal{F}_n$ be a filtration, i.e., an increasing sequence of σ -fields. A sequence X_n is said to be adapted to $\mathcal{F}_n$ if $X_n \in \mathcal{F}_n$ for all n . If X_n is a sequence with

1. $\mathbb{E}|X_n| < \infty$,
2. X_n is adapted to $\mathcal{F}_n$,
3. $\mathbb{E}[X_{n+1} | \mathcal{F}_n] = X_n$ for all n ,

then X_n is said to be a martingale (with respect to $\mathcal{F}_n$). If in the last definition, $=$ is replaced by $\leq$ or $\geq$, then X_n is said to be a supermartingale or submartingale, respectively.

Definition A.4 (Stopping time). A random variable N is said to be a stopping time if $\{N = n\} \in \mathcal{F}_n$ for all $n < \infty$, i.e., the decision to stop at time n must be measurable with respect to the information known at that time.

Theorem A.5 (Martingale convergence theorem, Theorem 4.2.11 of Durrett [4]). *If X_n is a submartingale with $\sup \mathbb{E}X_n^+ < \infty$, then as $n \rightarrow \infty$, X_n converges a.s. to a limit X with $\mathbb{E}|X| < \infty$.*

Theorem A.6 (Formal version of Theorem 3.2).

1. $H_\tau(\hat{\phi}_{\text{SDPO}}, \phi_0; \hat{\theta}_{\text{SFT}})$ is a stopping time.
2. Assume the following two conditions hold.

$$(i) \sup \mathbb{E}_{\tau_h \setminus \{s_1\} \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | s_1)} r_{\hat{\phi}_{\text{SDPO}}}^+(\tau_h) < \infty \text{ for any given } s_1 \sim \mathcal{D},$$

(ii) for any s_{h+1} , $R_{\hat{\theta}_{\text{SFT}}}$, V_{ϕ_0} and $V_{\hat{\phi}_{\text{SDPO}}}$ satisfy that

$$D_{\text{KL}}(R_{\hat{\theta}_{\text{SFT}}}(\cdot | s_h) || V_{\phi_0}(\cdot | s_h)) \geq D_{\text{KL}}(R_{\hat{\theta}_{\text{SFT}}}(\cdot | s_h) || V_{\hat{\phi}_{\text{SDPO}}}(\cdot | s_h)),$$

and

$$D_{\text{KL}}(R_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h) || V_{\phi_0}(\cdot | t_h)) \geq D_{\text{KL}}(R_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h) || V_{\hat{\phi}_{\text{SDPO}}}(\cdot | t_h))$$

Then $H_\tau(\hat{\phi}_{\text{SDPO}}, \phi_0; \hat{\theta}_{\text{SFT}})$ is finite with probability 1.

Remark A.7 (Discussion of the conditions in Theorem A.6). Condition (i) assumes that the verifier is finite for paths generated by reasoner $R_{\hat{\theta}_{\text{SFT}}}$ in expectation. This can easily be achieved when the verifier is finite.

Condition (ii) can be interpreted as assuming that the distance between $R_{\hat{\theta}_{\text{SFT}}}$ and V_{ϕ_0} is greater than the distance between $R_{\hat{\theta}_{\text{SFT}}}$ and $V_{\hat{\phi}_{\text{SDPO}}}$. This assumption is reasonable, as both $R_{\hat{\theta}_{\text{SFT}}}$ and $V_{\hat{\phi}_{\text{SDPO}}}$ align with human preferences and therefore tend to be closer to each other.

Proof of Theorem A.6. 1. Let $\mathcal{F}_h$ be the smallest σ -field generated by s_{h+1} , i.e., $\mathcal{F}_h = \sigma(s_{h+1})$. Observe that by the definition of the stopping rule as shown in equation (8), $\{H_\tau(\hat{\phi}_{\text{SDPO}}, \phi_0; \hat{\theta}_{\text{SFT}}) = h\}$ is determined by s_{h+1} . Hence $\{H_\tau(\hat{\phi}_{\text{SDPO}}, \phi_0; \hat{\theta}_{\text{SFT}}) = h\} \in \mathcal{F}_h$, which implies that $H_\tau(\hat{\phi}_{\text{SDPO}}, \phi_0; \hat{\theta}_{\text{SFT}})$ is a stopping time.

2. By the definition of $r_{\hat{\phi}_{\text{SDPO}}}$, we have

$$\begin{aligned} & \mathbb{E} \left[r_{\hat{\phi}_{\text{SDPO}}}(s_{h+1}) \mid s_h \right] \\ &= r_{\hat{\phi}_{\text{SDPO}}}(s_h) + \mathbb{E}_{t_h \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | s_h), a_h \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h), o_h = f_{a_h}(t_h)} \left[\log \frac{V_{\hat{\phi}_{\text{SDPO}}}(t_h | s_h)}{V_{\phi_0}(t_h | s_h)} + \log \frac{V_{\hat{\phi}_{\text{SDPO}}}(a_h | t_h)}{V_{\phi_0}(a_h | t_h)} \right]. \end{aligned}$$

Observe that condition (ii) implies that

$$\begin{aligned} & \mathbb{E}_{t_h \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | s_h)} \left[\log \frac{V_{\hat{\phi}_{\text{SDPO}}}(t_h | s_h)}{V_{\phi_0}(t_h | s_h)} \right] \\ &= D_{\text{KL}}(R_{\hat{\theta}_{\text{SFT}}}(\cdot | s_h) || V_{\phi_0}(\cdot | s_h)) - D_{\text{KL}}(R_{\hat{\theta}_{\text{SFT}}}(\cdot | s_h) || V_{\hat{\phi}_{\text{SDPO}}}(\cdot | s_h)) \\ &\geq 0, \end{aligned}$$

and

$$\begin{aligned} & \mathbb{E}_{a_h \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h)} \left[\log \frac{V_{\hat{\phi}_{\text{SDPO}}}(a_h | t_h)}{V_{\phi_0}(a_h | t_h)} \right] \\ &= \mathbb{E}_{a_h \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h)} \left[\log \frac{V_{\hat{\phi}_{\text{SDPO}}}(a_h | t_h)}{R_{\hat{\theta}_{\text{SFT}}}(a_h | t_h)} \right] + \mathbb{E}_{a_h \sim R_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h)} \left[\log \frac{R_{\hat{\theta}_{\text{SFT}}}(a_h | t_h)}{V_{\phi_0}(a_h | t_h)} \right] \\ &= D_{\text{KL}}(R_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h) || V_{\phi_0}(\cdot | t_h)) - D_{\text{KL}}(R_{\hat{\theta}_{\text{SFT}}}(\cdot | t_h) || V_{\hat{\phi}_{\text{SDPO}}}(\cdot | t_h)) \\ &\geq 0, \end{aligned}$$

hence $\mathbb{E} \left[r_{\hat{\phi}_{\text{SDPO}}}(s_{h+1}) \mid s_h \right] \geq r_{\hat{\phi}_{\text{SDPO}}}(s_h)$, which indicates that $r_{\hat{\phi}_{\text{SDPO}}}(s_h)$ is a submartingale.

Then combining condition (i) and Martingale convergence theorem A.5, $r_{\hat{\phi}_{\text{SDPO}}}(s_h)$ converges a.s. to a limit $\tilde{r}(s_1)$ as $h \rightarrow \infty$. Hence for $\epsilon > 0$, with probability 1, there exists a $H_\epsilon(s_1)$ such that $|r_{\hat{\phi}_{\text{SDPO}}}(s_{h+1}) - r_{\hat{\phi}_{\text{SDPO}}}(s_h)| < \epsilon$ when $h > H_\epsilon(s_1)$. Therefore, $H_\tau(\hat{\phi}_{\text{SDPO}}, \phi_0; \hat{\theta}_{\text{SFT}}) \leq H_\epsilon(s_1)$ with probability 1, as desired.

□

B Experimental Details

Table 4: **Hyperparameters for training Qwen2-VL-7B-Instruct & Qwen2.5-VL-7B-Instruct & LLaMA-3.2-11B-Vision-Instruct models**

Hyperparameter	Value
LoRA Rank	8
LoRA α	16
LoRA Dropout	0
LoRA Target	all
GPU	$8 \times$ NVIDIA A800
Per Device Train Batch Size	1
Gradient Accumulation Steps	1
Warmup Ratio	0.03
Learning Rate	$3e-5$
Learning Rate Scheduler	Cosine
Unfreeze Vision Tower	False
Number Train Epoch	1
Max Gradient Norm	1.0
bf16	True
Cut Off Length	65536

C Examples

C.1 VTS-DPO Dataset Examples

DPO EXAMPLES

Chosen



Q: What team hat is he wearing?

Reasoner

To determine the team hat, I will first use the grounding_action to locate the hat in the image. Then, I will zoom in on the hat to get a clearer view of any logos or text that might indicate the



Action : `grounding_action`
{`"image_index": 0, "text": ["hat"]`}

Reasoner

The hat has been located, and there seems to be a logo visible on it. I will now use the `zoomin_action` to focus on the logo for better clarity.



Action : `zoomin_action` {`"image_index": 2, "bounding_box": {"x_min": 171, "y_min": 182, "x_max": 272, "y_max": 260, "padding": 0.05}`}

Reasoner

The zoomed-in image shows a clear view of the logo on the hat. The logo appears to be 'NY', which is commonly associated with the New York Yankees baseball team. To confirm, I will use the OCR action to extract the text from

Action : `ocr_action` {`"image_index": 3, "engine": "easyocr" }`}

Reasoner

The OCR did not detect any text, likely due to the stylized nature of the logo. However, the visual appearance of the logo strongly resembles the 'NY' logo of the New York Yankees. Based on this observation, I can conclude that the hat is a New York Yankees hat.

Final Result

✓ The hat is a New York Yankees hat.

Rejected



Q: What team hat is he wearing?

Reasoner

To determine the team hat, I'll first try to identify the logo on the cap using the grounding action. This will help me recognize the team associated with the logo.



Action : `grounding_action`
{`"image_index": 0, "text": ["logo"]`}

Reasoner

Wrong action

I will now use the `visual_search_action` to thoroughly scan the image for any logos or text that could indicate the team. This will ensure I don't miss any details



Action : `visual_search_action` {`"image_index": 0, "objects": ["logo", "text"]`}

Reasoner

The visual search action highlighted some areas but didn't clearly identify the team. I will now use the OCR action to extract any text from the highlighted regions to see if it reveals the team name.

Action : `ocr_action` {`"image_index": 2, "engine": "easyocr" }`}

Reasoner

The OCR action did not yield clear text related to the team. Given the lack of clear identification, I will conclude that the team cannot be determined from the image.

Final Result

✗ Unknown

Figure 3: We present an example of the generated DPO data.

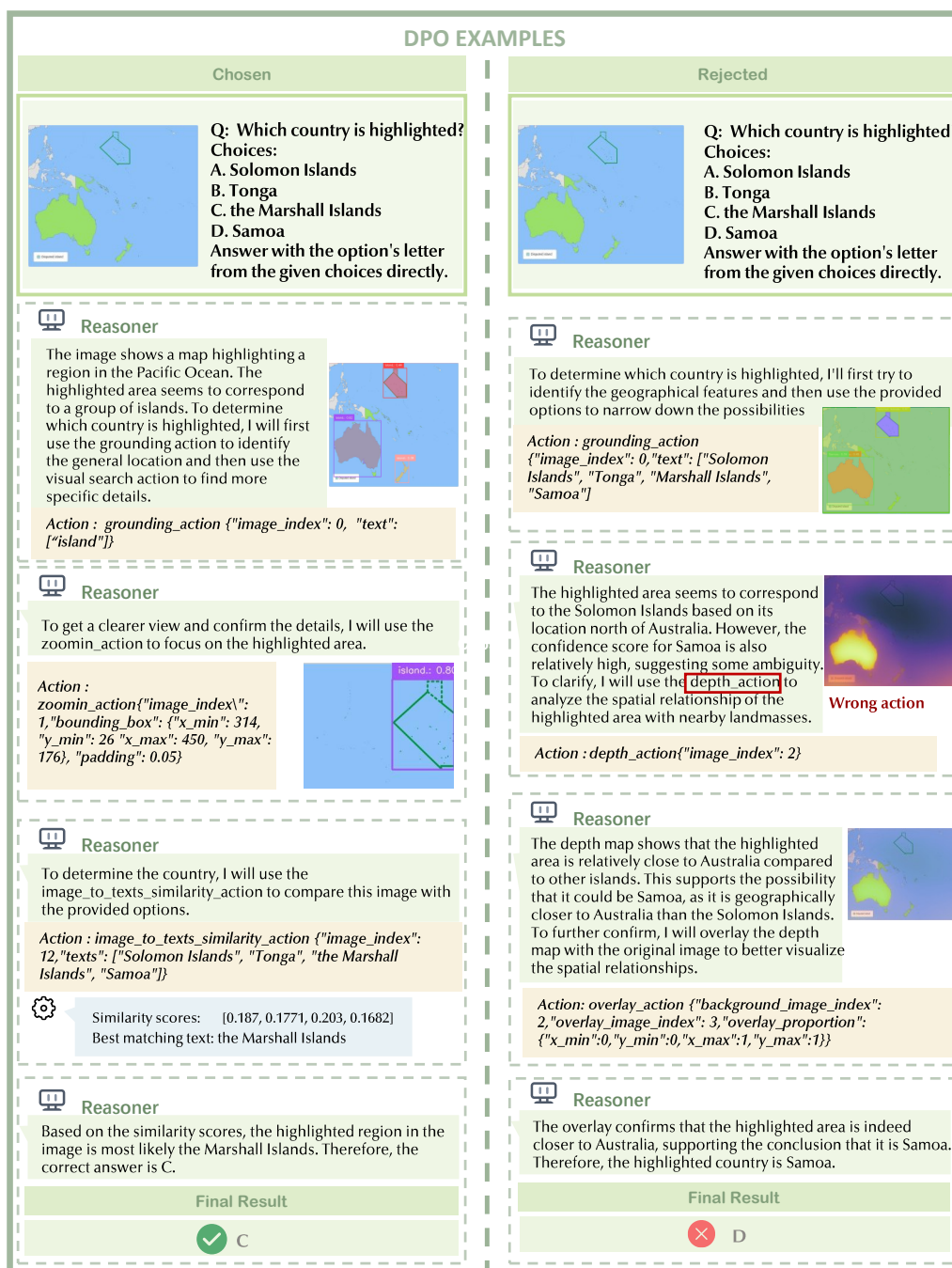


Figure 4: We present additional examples of the generated DPO data.

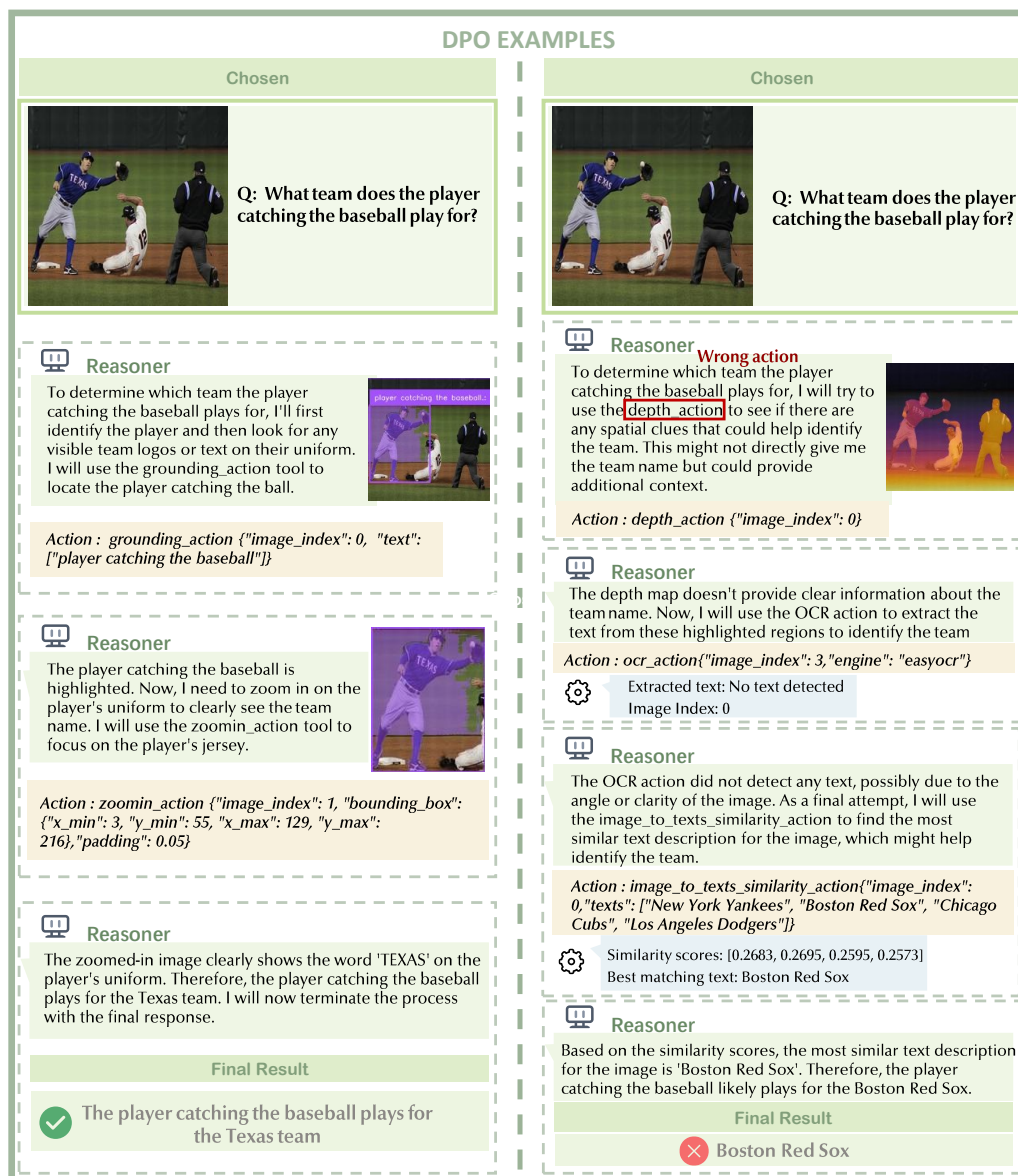


Figure 5: We present additional examples of the generated DPO data.

C.2 VTS-SFT Dataset Examples

SFT EXAMPLES



Q: What is the logo in the image?

Reasoner

To identify the logo, I will first use the OCR tool to extract any text present in the image. This will help me determine if there is a logo or brand name visible.

Action : ocr_action {"image_index": 0, "engine": "easyocr"}

Extracted text: YBE, Rs55, 8 K^rjF
Image Index: 0

Reasoner

The extracted text does not seem to include a recognizable logo or brand name. Next, I will use the grounding_action tool to locate any graphical elements that could represent a logo in the image.

Action : grounding_action {"image_index": 0, "text": ["logo"]}



Reasoner

The grounding action identified a potential logo area. Now, I will use the crop action to focus on the logo region for better clarity and further analysis.

Action : crop_action {"image_index": 2, "bounding_box": {"x_min": 30, "y_min": 339, "x_max": 253, "y_max": 562}, "padding": 0.05}



Reasoner

The cropped image visually suggests a logo resembling 'Baskin Robbins'. To confirm, I will use the image_to_texts_similarity_action to compare the cropped image with text descriptions related to popular ice cream brands.

Action : image_to_texts_similarity_action {"image_index": 3, "texts": ["Baskin Robbins", "Ben & Jerry's", "Haagen-Dasz"]}

Similarity scores: [0.3945, 0.2605, 0.2051]
Best matching text: Baskin Robbins

Reasoner

Based on the similarity scores, the logo in the image is most likely associated with 'Baskin Robbins'.

Final Result

The logo in the image is Baskin Robbins.



Q: Given an image of an animal, identify the kind of animal in the image. The picture could be of more popular animals that are visible around zoos or are sometimes domesticated at home. They could also sometimes be found in the wild.

Reasoner

To identify the kind of animal in the image, I will first use the grounding_action tool to locate any animals mentioned in the options. Since the image seems to contain elephants, I'll start by searching for 'elephant'.

Action : grounding_action {"image_index": 0, "text": ["elephant"]}



Reasoner

The grounding_action tool has successfully identified several elephants in the image. Now, I will use the depth_action tool to understand the spatial relationship of these elephants with respect to the camera and other objects in the scene.

Action : depth_action {"image_index": 2}



Reasoner

The depth map shows that the elephants are relatively close to the camera compared to the distant mountains. Now, I will use the overlay_action to combine the depth map with the original image to better visualize the spatial arrangement of the elephants and their surroundings.

Action : overlay_action {"background_image_index": 2, "overlay_image_index": 3, "overlay_proportion": {"x_min": 0, "y_min": 0, "x_max": 1, "y_max": 1}}



Reasoner

The overlaid image confirms the presence of elephants near the water and structures. Based on the visual evidence and the successful identification of elephants in previous steps, the correct answer is clear.

Final Result

elephant

Figure 6: We present examples of the SFT data.

<https://doi.org/10.52202/085713-2504>

83051



Figure 7: We present examples of the SFT data.



Figure 8: We present examples of the SFT data.

C.3 Benchmark Examples



Figure 9: We demonstrate examples of reasoning on benchmark data using VTS-V.



Figure 10: We demonstrate examples of reasoning on benchmark data using VTS-V.



Figure 11: We demonstrate examples of reasoning on benchmark data using VTS-V.

D Broader Impact

Our work presents a foundational framework for enhancing visual reasoning via Visual Token Scaling with Verification (VTS-V). While not tied to a specific deployment, the technology could pose risks if misused. For instance, improved fine-grained image understanding may be applied in surveillance or disinformation pipelines, raising concerns about privacy, fairness, and potential for malicious use. To mitigate such risks, we recommend responsible release practices (e.g., usage restrictions, auditing tools, or gated access) and encourage further research on bias and misuse detection. Though the societal impact is limited at this stage, we acknowledge its importance as the technology evolves.

E Inference-Time Overhead Analysis

This section provides a detailed quantitative analysis of the inference-time overhead introduced by our VTS-V framework, including latency, memory usage, and the number of intermediate tokens generated during dynamic reasoning.

Our method employs iterative reasoning steps composed of two main components:

- **Verifier calls:** Require only prefill computation (forward pass over the input context to generate stop/continue decisions)
- **Reasoner calls:** Involve autoregressive decoding to produce tool instructions

The iterative interaction follows this structured process:

```
context = input_prompt: Q (image, text_question)
while verifier(context) == 'continue':
    current_output = reasoner(context)
    physical_feedback = tool_using_API(current_output)
```

We define the total inference latency as:

$$\text{Latency} = \sum_{i=1}^H \left(\frac{C_i}{P_t} + \frac{O_i}{D_t} \right) \quad (15)$$

Where:

- H : Total number of reasoning steps
- C_i : Number of input tokens at step i (verifier prefill)
- O_i : Number of generated tokens at step i (reasoner decode)
- P_t : Model's prefill throughput (tokens/sec)
- D_t : Model's decode throughput (tokens/sec)

For Qwen2.5-VL-7B-Instruct with VTS-V, we observe the following characteristics:

- Average steps (H): 3.1
- Average verifier prefill tokens: ~ 10058.0 per step
- Average reasoner decode tokens: ~ 95.3 per step
- Throughput on 80GB A800 GPUs (float16):
 - $P_t \approx 18493.6$ tokens/s
 - $D_t \approx 231.8$ tokens/s

The total latency is calculated as:

$$\text{Total Latency} \approx 3.1 \times \left(\frac{10058.0}{18493.6} + \frac{95.3}{231.8} \right) \approx 2.96 \text{ seconds}$$

Memory requirements for our framework include:

- With `max_model_len=65536` and `dtype=float16`, models can be deployed within 32GB through vLLM or SGLang on 80GB A800 GPUs
- All tool models (including GroundingDINO, Depth-Anything, etc.) collectively require approximately 23 GB of additional memory during inference

We observe a clear tradeoff between reasoning depth and computational cost, which is justified by significant accuracy gains (e.g., +7.58% average improvement on BLINK for Qwen2.5-VL-7B-Instruct). To mitigate latency for practical deployment, our framework supports several optimization strategies, such as parallel verifier batching, early stopping, and adaptive step truncation via ϵ -threshold.

This analysis demonstrates that while our framework introduces additional computational overhead, the resulting accuracy improvements justify this cost, and practical deployment is feasible with appropriate optimizations.

F Extended Evaluation on Broader Benchmarks

This section presents extended evaluation results of our VTS-V framework on additional benchmarks, providing a more comprehensive assessment of its generalization capabilities across diverse visual reasoning tasks.

F.1 Experimental Setup

We conducted additional experiments evaluating GPT-4o augmented with our VTS-V framework on three challenging benchmarks: V*Bench [27], MMStar [3], and MathVista [18]. For comparison, we included the following baseline methods:

- **GPT-4o**: The base model without additional reasoning mechanisms
- **GPT-4o + CoT**: GPT-4o with Chain-of-Thought prompting [26]
- **GPT-4o + Sketchpad**: GPT-4o with Visual Sketchpad framework [11]

F.2 Results and Analysis

Table 5 summarizes the performance comparison across different benchmarks:

Table 5: Performance comparison of GPT-4o variants on broader benchmarks. Best results are in **bold**.

Method	V*Bench	MMStar	MathVista
GPT-4o	60.73	58.40	63.80
GPT-4o + CoT	60.73	56.47	66.50
GPT-4o + Sketchpad	71.20	60.33	59.40
GPT-4o + VTS-V (Ours)	73.82	65.80	65.90

Key observations from these extended evaluations:

- Our VTS-V framework achieves state-of-the-art performance on both V*Bench and MMStar benchmarks, with absolute improvements of **13.09%** and **7.40%** over the baseline GPT-4o, respectively. These results demonstrate the effectiveness of our verifier-guided visual token scaling approach for complex visual reasoning tasks.
- On the MathVista benchmark, our method outperforms the base GPT-4o by **2.10%** and achieves competitive performance compared to GPT-4o + CoT, with only a **0.60%** difference from the top-performing method. This demonstrates the versatility of our approach across mathematical reasoning in visual contexts.
- The consistent improvements across diverse benchmarks highlight the generalization capability of our framework, which effectively enhances visual reasoning without task-specific adaptations.

G Analysis of Performance Differences Between GPT-4o and Open-Source Models

This section analyzes the performance differences observed when applying VTS-V to GPT-4o versus open-source models without fine-tuning. While GPT-4o shows consistent improvements with VTS-V without additional training, open-source models like Qwen2-VL and LLaMA-3.2-Vision initially experience performance degradation. This discrepancy stems from several key factors:

- **Tool-Using Priors**: GPT-4o has extensive exposure to tool-augmented workflows during pretraining, enabling it to handle structured reasoning formats (e.g., Plan → Action → Tool call) naturally. Open-source models lack this prior and struggle to generate valid tool-use commands without fine-tuning.
- **Instruction Following Robustness**: GPT-4o’s alignment tuning (e.g., RLHF) allows it to interpret multi-turn instructions and verifier signals effectively. Open-source models require fine-tuning to learn how to respond to verifier guidance correctly.

- **Empirical Recovery via Fine-Tuning:** After supervised fine-tuning on our VTS-SFT dataset, open-source models not only recover but exceed baseline performance (e.g., Qwen2VL-7B improves from 48.01% to 52.19%), demonstrating the dataset’s role in enabling tool-use behaviors.

This analysis highlights the importance of our VTS-SFT dataset for adapting open-source models to verifier-guided reasoning, while confirming the framework’s generality with state-of-the-art models like GPT-4o.

H Visual Token Scaling Analysis with Optimal-Stopping Verifier

This section presents experimental results demonstrating the relationship between multi-step reasoning performance and visual token scaling, highlighting our framework’s dynamic behavior and the effectiveness of the optimal-stopping verifier.

Table 6: Visual token scaling performance for GPT-4o on BLINK Counting task

Visual & Text Tokens Per Sample	GPT-4o BLINK Counting Accuracy
550	49.17%
1100	58.33%
1650	67.50%*
2200	64.16%
2750	65.83%

Table 7: Visual token scaling performance for open-source models on BLINK subtasks

Visual & Text Tokens	Qwen2.5-VL Fun.Corr.	Qwen2-VL Depth	LLaMA-3.2 Spatial
600	19.23%	56.45%	62.94%
1200	23.85%	60.48%*	65.03%
1800	25.38%	58.87%	69.23%*
2400	30.00%*	57.26%	67.83%
3000	26.15%	58.06%	67.13%

The tables illustrate the scaling relationship between reasoning accuracy and token usage across different models. On average, GPT-4o generates approximately 50 text tokens per reasoning step, while our trained open-source models generate around 100 text tokens per step, with both processing approximately 100 visual tokens through dynamic image operations per step.

Key observations from the scaling analysis:

- **Non-monotonic scaling:** Accuracy improves with increased visual tokens initially, but may decline with excessive reasoning steps, indicating diminishing returns.
- **Verifier-guided optimal stopping:** Our VTS-V verifier consistently identifies the best stopping points (marked by *), balancing reasoning depth and computational efficiency.
- **Robust performance:** Accuracy remains substantially higher than under-reasoning baselines even beyond optimal steps, demonstrating the benefits of sufficient visual grounding.

These results validate our verifier’s ability to dynamically scale visual tokens while maintaining optimal reasoning performance across different model architectures and task types.

I Robustness and Generalization Analysis of the Verifier

This section provides additional analysis of the verifier’s robustness and generalization capabilities, addressing its performance on unseen domains and tasks beyond the training distribution.

The VTS-DPO dataset is constructed from the comprehensive LLaVA-OneVision (LLaVA-OV) dataset, which spans over 80 distinct vision-language task types. Our curated DPO dataset comprises 301,028 preference pairs that cover a diverse range of visual reasoning domains, ensuring broad coverage and representation.

Table 8: VTS-DPO dataset composition across different domains

Domain Category	# Tasks	# Examples
General VQA	25	135,804
Document/Chart/Screen	18	100,681
Math/Logical Reasoning	15	52,049
OCR/Text-grounded Tasks	6	12,494

The dataset encompasses diverse visual reasoning capabilities including document understanding, scientific diagram interpretation, mathematical problem solving, chart question answering, optical character recognition, and general commonsense visual question answering.

Importantly, all evaluation benchmarks used in our experiments (BLINK, V*Bench, MMStar, and MathVista) are held-out from the training data, ensuring that the verifier is tested on completely unseen reasoning tasks and domains. The consistent performance improvements observed across these diverse benchmarks demonstrate the verifier’s strong generalization capabilities beyond its training distribution.

The comprehensive coverage of VTS-DPO, combined with the held-out evaluation protocol, validates the robustness of our verifier training approach and its applicability to novel visual reasoning scenarios.

J Separate Reasoner and Verifier

This section explains our design decision to maintain separate reasoner and verifier components rather than employing a single unified model. While a unified approach offers potential efficiency benefits, our analysis demonstrates that a decoupled architecture provides superior performance for multi-modal, tool-augmented reasoning tasks.

Reasoning and verification represent fundamentally distinct capabilities: the reasoner must generate executable tool-using action plans conditioned on complex multi-modal contexts, while the verifier must evaluate the overall quality and utility of reasoning steps. Empirical observations show that when both responsibilities are assigned to a single model, performance typically deteriorates as the model struggles to maintain both generative planning and discriminative evaluation capabilities.

Experimental results provide strong validation for the separated architecture. As shown in Table 9, adding a standalone verifier module to GPT-4o yields substantial performance gains:

Table 9: Performance comparison of GPT-4o with and without separate verifier on BLINK benchmark

Model	Avg Accuracy (BLINK)
GPT-4o	62.25%
GPT-4o + Verifier	69.15% (+6.90%)

These improvements are particularly pronounced on tasks requiring structured multi-step reasoning, with gains of +18.33% on Counting and +12.91% on Functionally-Correlated tasks. Beyond performance benefits, the separated architecture offers practical advantages including model-agnostic verifier deployment across architectures (GPT-4o, Qwen, LLaMA), reduced fine-tuning costs, and enhanced extensibility for new tasks and domains. This analysis confirms that the separated reasoner-verifier design provides both performance benefits and practical advantages for complex visual reasoning tasks.

K Interpretability of Reasoning Traces

This section clarifies our claim regarding the improved interpretability of reasoning traces generated by our VTS-V framework. We define interpretability in this context as the explicit, grounded, and stepwise nature of the reasoning process that makes the model’s decision-making transparent and verifiable.

Traditional vision-language models perform latent, one-shot image encoding and generate answers based on static visual token embeddings, offering limited insight into their reasoning process. In contrast, our framework produces multi-turn reasoning traces composed of explicit tool-use steps, where each turn includes:

- A verbalized subgoal or planning statement
- A specific tool invocation with parameters
- Visual feedback or observation from tool execution

These intermediate outputs are grounded in actual visual content through operations such as region cropping, OCR text extraction, depth map generation, and spatial analysis. This structured output format provides a transparent reasoning trajectory that allows both human users and automated systems to inspect the decision process at each step.

Empirical evidence supports this improved interpretability. Across multiple benchmarks, our models generate longer and more structured reasoning trajectories compared to vanilla baselines. For instance, the average reasoning steps increase from approximately 1.5 (GPT-4o baseline) to 3.2 (our VTS-V framework), with diverse tool usage patterns including grounding actions, depth analysis, OCR operations, and region cropping.

Examples of these interpretable reasoning traces are visible in Figure 1 and Appendix C.3 of the main paper, demonstrating how models produce dynamic reasoning involving tool selection and verification-guided iteration. While future work could include manual evaluations of reasoning quality, our current framework already provides verifiable, human-readable sub-decisions that represent a significant advancement beyond traditional black-box VLM inference.

L Ablation Study on the Reasoner Module

This section presents an ablation study evaluating the specific contribution of the reasoner module within our VTS-V framework. We compare two configurations: models using verifier guidance without the trained reasoner (tool actions generated directly by the base model) versus our full VTS-V framework with both trained reasoner and verifier components.

Table 10: Performance comparison with and without the reasoner module

Model Variant	V*Bench	MMStar	MathVista
Qwen2.5VL-7B + Verifier (w/o Reasoner)	67.54	55.93	7.80
Qwen2.5VL-7B + VTS-V (w/ Reasoner)	75.13	57.93	23.52
Qwen2VL-7B + Verifier (w/o Reasoner)	50.80	44.29	20.24
Qwen2VL-7B + VTS-V (w/ Reasoner)	66.67	55.26	23.80

The results demonstrate that incorporating the trained reasoner significantly improves performance across all benchmarks. Without the reasoner module, models struggle to perform reliable multi-step reasoning, particularly on complex tasks like MathVista where performance degrades substantially.

The addition of the reasoner consistently produces large performance gains, with improvements of up to +15.72 points on MathVista for Qwen2.5VL-7B and +15.87 points on V*Bench for Qwen2VL-7B. These findings highlight the critical importance of the reasoner module in executing valid tool-use actions and effectively following verifier guidance during multi-step reasoning processes.

M Comparison with Dynamic Visual Attention Methods

This section provides a comparative analysis between our VTS-V framework and prior work on dynamic visual attention, highlighting key distinctions in methodology and capabilities.

Our framework introduces a general, verifier-guided inference-time reasoning process based on a Markov Decision Process (MDP) formulation, enabling multi-step refinement of visual understanding through a flexible set of visual tools. This approach differs fundamentally from existing methods in several important aspects:

- **LMEye**: While LMEye allows large language models to request new visual evidence, it lacks a structured action space and formal termination control. Our approach models perception as MDP-driven tool selection with a multi-step DPO-trained verifier that dynamically determines when to terminate reasoning.
- **InstructBLIP**: This method enhances initial visual encoding through instruction tuning but performs single-pass perception. In contrast, our framework supports iterative, adaptive reasoning during inference, allowing models to invoke diverse tools (OCR, zoom, grounding) based on evolving contextual understanding.
- **V***: Primarily designed for recursive visual search to locate objects in high-resolution images, V* operates with a binary search objective. Our framework addresses a broader range of tasks including mathematical reasoning, OCR, and chart understanding, with termination based on verifier-guided reasoning quality rather than search completion.

Our core innovations distinguish VTS-V from these approaches:

1. **Inference-time visual token scaling**: Dynamically adjusts which image regions receive detailed processing through tool invocation, rather than using fixed input-time processing.
2. **MDP-based multi-step reasoning**: Formally defines each state-action step to support structured perception and tool-based decision-making with theoretical guarantees.
3. **Verifier-guided termination**: Employs a learned verifier trained via multi-step DPO to provide step-wise feedback and enforce bounded reasoning (Lemma 3.2), improving both efficiency and robustness.

These innovations position VTS-V as a more comprehensive and theoretically grounded approach to dynamic visual reasoning compared to existing methods.

N Novelty Analysis and Comparison with Prior Work

In this section, we provide a detailed analysis of the novelty of our VTS-V framework and its distinctions from related works. While our method shares some conceptual similarities with prior research in iterative reasoning, reasoner-verifier separation, and tool use, it introduces several key innovations that advance the state of the art in multi-step visual reasoning:

- **MDP-based Reasoning Framework for Dynamic Visual Token Scaling**: Unlike V* [27] which primarily focuses on task-specific visual cropping, we formulate multi-step visual reasoning as a *Markov Decision Process* over a diverse and extensible set of visual tools (e.g., OCR, grounding, depth, segmentation). This formulation enables dynamic visual token scaling—allowing the model to adaptively focus on different regions or modalities at each reasoning step—across a wide range of tasks and model architectures.
- **Step-wise Verifier with Theoretical Guarantees**: While V-STaR [9] employs a verifier to rank final answers, our verifier operates **at each reasoning step**, providing real-time guidance and enabling early termination with theoretically bounded steps (Theorem 3.2). This ensures both interpretability and computational efficiency, a feature not present in prior verifier-based methods.
- **First Dataset for Multi-Step Visual Tool Reasoning**: Existing tool-use datasets (e.g., from Visual Program Distillation [11]) do not support multi-turn visual reasoning with rich intermediate states. We introduce VTS-SFT and VTS-DPO, the first datasets designed for fine-grained tool selection and verifier training over more than 80 diverse vision-language tasks, enabling robust generalization and grounding.
- **Generalization Without Fine-Tuning**: Our framework generalizes effectively even to closed-source models like GPT-4o without additional fine-tuning, improving performance by **+6.9%** on the BLINK benchmark. This demonstrates the flexibility and model-agnostic nature of our approach.

In summary, while our work builds upon prior art, it integrates these ideas into a unified, theoretically sound, and empirically validated framework that supports multi-step, tool-augmented, verifier-guided visual reasoning. We believe this represents a meaningful step forward in enabling fine-grained and interpretable reasoning in multimodal systems.