OMNIVL-GUARD PRO: A TOOL-AUGMENTED AGENT FOR OMNIBUS VISION-LANGUAGE FORENSICS

Anonymous authors
Paper under double-blind review



Figure 1: This work tackles omnibus vision-language forgery detection, grounding, and segmentation (left). In this unified setting, models relying on parametric memory alone cannot handle manipulations that require external clues or fine-grained inspection. In response, we propose OmniVL-Guard Pro, a tool-augmented agent trained with Checker-Guided Agentic RL (CGARL), achieving evidence-grounded reasoning and consistent gains over SFT across all tasks (right).

ABSTRACT

Existing vision-language forgery detection and grounding methods operate under a closed-world paradigm, assuming verification can be completed by the model alone. However, self-contained MLLMs are constrained by finite parametric knowledge, static training corpora, and limited perceptual resolution, creating a practical ceiling in dynamic open-world forensics—particularly for real-time event verification requiring external clues and forgery segmentation demanding fine-grained scrutiny of local manipulations. To address these limitations, we shift from scaling up the self-contained model toward reaching beyond it. We propose **OmniVL-Guard Pro**, a tool-augmented agent that extends unified forensics from closed-world prediction to open-world clues-driven reasoning. OmniVL-Guard Pro integrates a tool environment spanning real-time event search, local cropping and zooming, edge-anomaly screening, face detection, video frame extraction, and SAM3-based segmentation. To generate high-quality tool-reasoning trajectories, we introduce **Tree-Structured Self-Evolving Tool Trajectory Generation**, which produces diverse trajectories through seed guidance, guider-free self-evolution, and weakly-hinted hard sample synthesis, yielding the Full-Spectrum Tool Reasoning (FSTR) dataset for training. We further propose **Checker-Guided Agentic Reinforcement Learning** (CGARL), which provides process-level supervision to penalize cases where the answer is correct but the reasoning is distorted. Extensive experiments demonstrate that OmniVL-Guard Pro achieves state-of-the-art performance across various tasks, and exhibits strong zero-shot generalization. The FSTR dataset and code for OmniVL-Guard Pro will be publicly released.

1 INTRODUCTION

The rapid evolution of generative AI has significantly lowered the barrier for creating manipulated content across diverse modalities to infiltrate real-world social media ecosystems at an unprecedented

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

pace (Haydarov et al., 2024; Li et al., 2024b; Abdelnabi et al., 2022). Consequently, forgery detection, manipulation localization, and forgery segmentation across primary modalities including images, videos, and text have garnered substantial attention (Shen et al., 2025; Liu et al., 2024). Despite recent advancements such as OmniVL-Guard (Shen et al., 2026), which establishes a unified framework for omnibus vision-language forgery detection and grounding, existing methods still mainly operate under a *closed-world* paradigm, which implicitly assumes that ***the verification process can be completed by the model itself.***

However, self-contained MLLMs (DeepSeek-AI, 2025; Zhang et al., 2025) remain constrained by finite parameters, static training corpora, and limited native perceptual resolution, which impose a practical ceiling in dynamic open-world forensics. This limitation is evident in two representative settings: Real-time Event Verification requires up-to-date external clues beyond parametric memory, while Forgery Segmentation demands fine-grained scrutiny of local boundaries, texture discontinuities, facial details, and semantic inconsistencies for pixel-level localization. Although scaling model size or training data could partially enhance these abilities, such a "scaling-up" strategy is costly and insufficient for continuously evolving real-world forgeries. We therefore shift from merely scaling up the self-contained model to reaching out beyond it, enabling the model to retrieve real-world knowledge and invoke specialized perception tools when necessary. In this sense, these two tasks jointly motivate an open-world forensic agent that ***dynamically acquires external knowledge and fine-grained visual artifacts to transcend the intrinsic limits of the base model***.

To this end, we first construct a tool-enhanced environment that covering key capabilities such as real-time event search, local cropping, and segmentation (Sec. 2). This environment enables the model to evolve from a self-contained predictor into an clues-seeking agent that actively gathers, examines, and integrates external clues.

With the tool environment as the base, the agent needs to decide when and which tools is invoked. Cultivating this ability requires high-quality tool-based reasoning trajectory data. Unlike pure-text Chain-of-Thought (CoT), tool-based reasoning trajectory simultaneously involves tool selection, parameter specification, observation understanding, and stopping decisions, making it substantially more difficult to construct. Even state-of-the-art MLLMs (Zhao et al., 2024; Team, 2025; Guo et al., 2025) still exhibit limited exploration quality in real tool environments, and thus cannot directly produce sufficiently high-quality training trajectories. Conversely, directly injecting GT or answer-aware cues would cause the model to organize tool usage around the known answer, thereby introducing hindsight bias (Shen et al., 2026). To address this challenge, we propose a **Tree-Structured Self-Evolving Forensic Tool Trajectory Generation** framework. In the seed stage, it efficiently constructs high-fidelity trajectories through tree expansion and branch pruning. It then gradually scales up the data through unguided self-evolution, while further strengthening long-tail hard samples with weak hints, thereby providing cold-start data for subsequent tool-augmented RL.

Taking the trajectory data, we finally train the model for detection, grounding and segmentation with the aid of external tools. However, merely teaching the model to invoke tools is still insufficient. The core of forgery detection, grounding, and segmentation lies not only in obtaining external clues/artifacts, but also in forming correct and consistent judgments based on such evidence. The model still exhibits a pseudo-success pattern where the ***"answer is accidentally correct but the process is distorted"***: although the final answer happens to be correct, the intermediate tool calls, observational evidence, and final judgment do not form a consistent and verifiable support relation, and such deviations are difficult for the model itself to identify. To tackle this problem, we further propose **Checker-Guided Agentic Reinforcement Learning** (CGARL). Specifically, we introduce a *Checker* to build a Multi-Agent system, which provides process-level supervision over intermediate reasoning, tool usage, and observational evidence, explicitly assessing whether they jointly support the final judgment. Through this mechanism, the model is encouraged to genuinely organize its reasoning around evidence, rather than remaining at the stage of merely knowing how to call tools without properly using the acquired evidence.

Built upon the above designs, we propose **OmniVL-Guard Pro**, an open-world tool-augmented Agent model. By integrating Tree-Structured Self-Evolving Forensic Tool Trajectory Generation with Checker-guided process-supervised RL, OmniVL-Guard Pro advances OmniVL-Guard from a closed-world unified forensic model into an open-world evidence-driven intelligent agent. Overall, our main contributions are summarized as follows:

(1) **Open world Agent for Vision-Language Forensics.** We propose OmniVL-Guard Pro, a tool-augmented Agent that performs forgery detection, grounding, and segmentation with external support across text, image, and video modalities, supporting real-time event verification and fine-grained forgery segmentation. We also construct a benchmark for real-time event verification.

(2) **FSTR Dataset.** We construct *Full-Spectrum Tool Reasoning* dataset, consisting of $\mathrm{FSTR}_{\mathrm{sft}}$ for cold-start SFT and $\mathrm{FSTR}_{\mathrm{rl}}$ for subsequent tool-augmented RL. $\mathrm{FSTR}_{\mathrm{sft}}$ contains high-fidelity tool-search trajectories, while $\mathrm{FSTR}_{\mathrm{rl}}$ supports subsequent tool-augmented RL.

(3) **Checker-Guided Agentic RL.** We propose Checker-Guided Agentic RL, which introduces process-level supervision beyond answer-level rewards to encourage evidence-consistent reasoning. It mitigates the pseudo-success pattern where the "answer is correct but the process is distorted," improving both Agent performance and reasoning-evidence consistency.

2 Tool-Enhanced Environment

To enable open-world evidence acquisition beyond intrinsic model capabilities, we construct a tool environment for unified vision-language forgery detection, grounding, and segmentation.

Real-time event search tool. This tool searches the web given a keyword query (e.g., time, person, location), returning the most relevant results and supporting clues for real-time fact verification.

Local cropping tool. Given normalized coordinates, this tool extracts the specified region as a new image context for further inspection. **Local zooming tool.** Given a candidate region, this tool returns an enlarged view for fine-grained inspection of boundary discontinuities, texture anomalies, local blur, and subtle artifacts.

Edge-anomaly screening tool scans the image using local edge density, Laplacian energy, and neighborhood discrepancy, and returns candidate regions for cropping and zooming. **Face detection tool** returns face bounding boxes based on InsightFace (Deng et al., 2021), supporting facial region inspection in portrait and person-centric media.

Video clip frame extraction tool. Given a candidate temporal segment, this tool extracts frames for further examination using the image-level tools above.

SAM3 segmentation tool runs SAM3 (Carion et al., 2025) to produce a pixel-level mask from model-derived prompts: bounding box, positive points inside the target, and negative points in the background.

3 Tree-Structured Self-Evolving Tool Trajectory Generation

High-quality tool-search trajectories are a critical prerequisite for the cold-start phase of tool-augmented reinforcement learning. However, constructing such data in unified vision-language scenarios faces a new **efficiency–bias dilemma**: on the one hand, if we directly rely on general-purpose MLLMs to freely explore in real tool environments, they often produce a large number of low-quality trajectories since stable tool-use capabilities have not yet been established; on the other hand, if GT or answer-aware cues are injected to force-guide the search process, hindsight bias will be introduced, causing the model to learn answer-conditioned tool invocation patterns rather than transferable evidence-seeking capabilities. To address this issue, we propose a **Tree-Structured Self-Evolving Tool Trajectory Generation** strategy. Overall, this strategy consists of four stages, as shown in Figure 2.

3.1 Source Data Collection

We collect source data covering text, images, videos, image–text, and text–video for detection, grounding, and segmentation; the complete dataset inventory is provided in Appendix D.4. For real-time event verification, we use RealFact (Appendix B). Finally, we partition all source data into three parts: $\mathcal{D}_s$ for SFT cold-start data construction, $\mathcal{D}_r$ for subsequent RL data construction, and $\mathcal{D}_t$ for in-domain testing. The selection of out-of-domain zero-shot test data is detailed in Sec. 5.

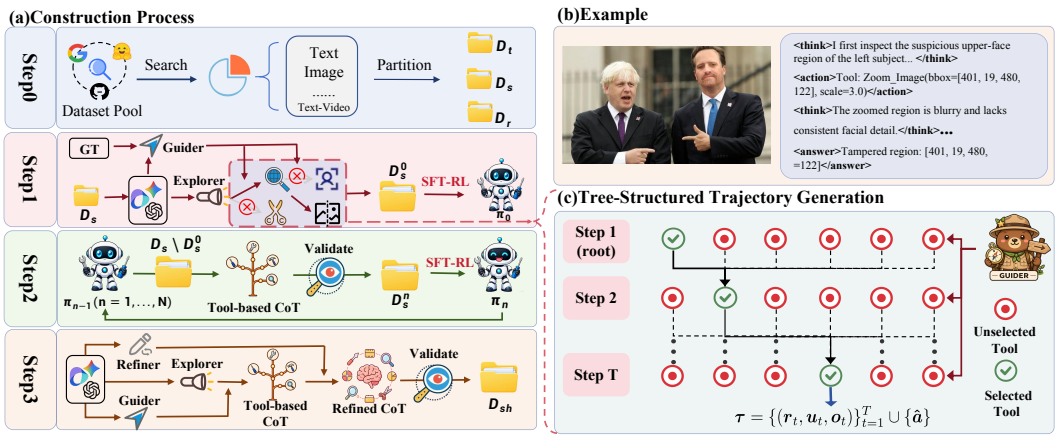


Figure 2: (a)The construction of FSTR_{sft}. (b) An example from FSTR_{sft}. (c) The process of Tree-Structured Trajectory Generation.

3.2 Tool Trajectory Seed Guidance

On a small-scale subset of $\mathcal{D}_s$, we first construct a high-fidelity seed set of tool-search trajectories. To this end, we define an MLLM expert pool $\mathcal{M} = \{\text{GPT5.4},\ \text{Gemini3.1},\ \text{Seed2.0}\}$. For each sample, we randomly select two distinct models from $\mathcal{M}$ to serve as the *Explorer* and the *Guider*, respectively. The *Explorer* is responsible for executing search in the real tool environment, while the *Guider*, with the final answer $a^\star$ injected, determines the validity of the tool attempt performed at the current step. Based on this design, seed trajectory generation is not a linear rollout along a single path, but rather a tree-structured search process composed of branch expansion and branch pruning, as shown in Figure 2(c).

Formally, a tool-search trajectory is denoted as

$$\tau = \{(r_t, u_t, o_t)\}_{t=1}^{T} \cup \{\hat{a}\}, \tag{1}$$

where r_t denotes the model's reasoning process at step t, which is formed based on previous tool-returned observations and determines the next tool to invoke as well as its parameters. u_t denotes the selected tool and its parameters at the current step, o_t denotes the observation returned by the tool, and $\hat{a}$ denotes the final prediction produced by the model at the end of the trajectory.

At step t, given the current history $\{(r_i, u_i, o_i)\}_{i=1}^{t-1}$, the *Explorer* expands the current node by sequentially attempting each tool in the tool set $\mathcal{T}$ (Sec. 2). For each candidate tool, the *Explorer* first generates the reasoning process r_t, which analyzes the previous observations and specifies both the tool to be invoked and its parameters, such as a search query, a crop box, or a zooming region. The corresponding tool invocation is denoted as u_t, and is executed in the real tool environment to obtain the returned observation o_t. The *Guider*, with the final answer $a^\star$ injected, then evaluates whether this candidate step (r_t, u_t, o_t) is helpful for solving the final task. If it is judged invalid, the current branch is immediately terminated, and the *Explorer* turns to the next candidate tool under the same history state; if it is judged valid, (r_t, u_t, o_t) is appended to the trajectory, and the resulting new state is further expanded in the next step following the same procedure.

After the *Explorer* completes the search and outputs the final prediction $\hat{a}$, we further conduct quality checking on the trajectory, requiring the final answer to be correct, the overall trajectory to be coherent, the intermediate reasoning and tool usage to support the final conclusion, and the trajectory to contain no obvious ineffective steps. The detailed criteria for the *Guider* judgment and the subsequent quality checking process are provided in Appendix C.1 C.2. All trajectories that pass the checking process are included in the seed set $\mathcal{D}_s^0$. Subsequently, we perform SFT on Qwen3VL-8B (Yang et al., 2025a) using $\mathcal{D}_s^0$, followed by RL on $\bar{\mathcal{D}}_r \subset \mathcal{D}_r$, yielding the initial policy π_0.

3.3 GUIDER-FREE SELF-EVOLVING EXPANSION

After obtaining the initial policy π_0, we further design a Guider-free self-evolving expansion mechanism to continuously scale up high-quality tool-augmented reasoning trajectories. Specifically, in the n-th iteration, we use π_{n-1} to perform multi-turn rollout reasoning on $\mathcal{D}_s \setminus \mathcal{D}_s^0$, producing candidate tool trajectories. Different from the seed stage, we remove the answer-aware Guider from the rollout process, so the current policy autonomously explores the tool environment without Guider-based step-level judgment or branch pruning. For each candidate trajectory, we apply the same checking and filtering procedure as in Sec. 3.2, retaining only samples whose final answers are correct, whose trajectories are globally self-consistent, whose tool usage and reasoning processes support the final conclusion, and which contain no obvious ineffective steps (Appendix C.2). All trajectories that pass the filtering process are combined with $\mathcal{D}_s^0$ to form the current-round dataset $\mathcal{D}_s^n$. We then re-initialize SFT from Qwen3VL-8B using $\mathcal{D}_s^n$, followed by RL on $\bar{\mathcal{D}}_r$, obtaining the n-th round policy π_n. We repeat this self-evolving process for $N = 4$.

3.4 WEAKLY-HINTED HARD TRAJECTORY SYNTHESIS

Although self-evolution can substantially improve the coverage of valid trajectories, standard rollout could still fail to produce any accepted trajectory for samples whose correct tool-use paths require sparse clues, precise local regions, or non-obvious search keywords. These persistently failed samples constitute the long-tail portion of the trajectory distribution. To complement the self-evolved data on such hard cases, we further design a weakly-hinted hard trajectory synthesis mechanism.

Specifically, for samples that consistently fail during the self-evolution stage, following Sec. 3.2, we randomly select two models from $\mathcal{M}$ to serve as *Explorer* and *Guider*, respectively. The *Explorer* sequentially attempts each tool in the tool set $\mathcal{T}$ at each step to generate candidate trajectories, while the *Guider*, with the final answer $a^\star$ injected, judges the tool attempt at the current step. Different from Sec. 3.2, the *Guider* not only determines whether the current tool selection is valid, but also provides usage hints for the tool, namely necessary parameters or local clues related to the current tool invocation, to narrow down the search space at the current step for the *Explorer*. For example, in image scenarios, the *Guider* provides a rough candidate region to assist cropping or zooming. Based on these weak hints, the *Explorer* performs tool search and generates candidate trajectories.

After obtaining the weakly-hinted candidate trajectories, we further randomly select another model from $\mathcal{M}$ to serve as the *Refiner*, which rewrites the reasoning process r_t at each step of the trajectory to eliminate explicit traces introduced by weak hints, preventing such answer-aware hints from contaminating the trajectory and inducing hint-dependent reasoning. We then apply the same quality checking procedure as in Sec. 3.2. All trajectories that pass the checking process are included in the hard-sample set $\mathcal{D}_{sh}$. The rewriting details are provided in Appendix C.4.

3.5 DATASET STATISTICS

Through the aforementioned process, we obtain the Full-Spectrum Tool Reasoning (FSTR) dataset. It consists of two parts: the tool-search trajectory set for cold-start SFT, $\mathrm{FSTR}_{\mathrm{sft}} = \mathcal{D}_s^N \cup \mathcal{D}_{sh}$; and the dataset for RL training, $\mathrm{FSTR}_{\mathrm{rl}} = \mathcal{D}_r \setminus \bar{\mathcal{D}}_r$. $\mathrm{FSTR}_{\mathrm{sft}}$ contains 69,323 samples, where $\mathcal{D}_{sh}$ accounts for approximately 17% and $\mathcal{D}_s^0$ accounts for approximately 8%; $\mathrm{FSTR}_{\mathrm{rl}}$ contains 81,087 samples. FSTR covers five core modalities, including text, image, video, text-image, and text-video, and supports full-spectrum vision-language classification tasks, omni-modal grounding tasks, image-modal segmentation tasks, and real-time event verification tasks.

4 CHECKER-GUIDED AGENTIC REINFORCEMENT LEARNING

After obtaining the cold-start tool trajectory data, we adopt a three-stage training strategy. Specifically, we first perform SFT on $\mathrm{FSTR}_{\mathrm{sft}}$ and then split $\mathrm{FSTR}_{\mathrm{rl}}$ into two disjoint subsets, $\mathcal{D}_{\mathrm{rl}}^1$ and $\mathcal{D}_{\mathrm{rl}}^2$. In the first stage, we conduct outcome-level RL on $\mathcal{D}_{\mathrm{rl}}^1$, yielding a tool-capable policy π_{ans}. In the second stage, we rollout π_{ans} on $\mathcal{D}_{\mathrm{rl}}^2$ to construct process supervision data and train the process discrimination model π_{checker}. The third stage further performs Checker-guided process-supervised RL on $\mathcal{D}_{\mathrm{rl}}^2$, jointly improving answer correctness, process consistency, and tool-use efficiency. The rationale behind this design is that effective Checker supervision requires a policy with stable tool-use

Table 1: Performance comparison on the in-domain test set $\mathcal{D}_t$. Deltas compare against the baselines listed in this table.

Method	Binary Classification (ACC)						Localization			Seg.
	Text	Image	Video	Text-Image	Text-Video	RealFact	Image (IoU)	Text (F1)	Video (tIoU)	(Dice)
Multi-modal Large Language Models										
Qwen3VL-235B (Yang et al., 2025a)	89.23	65.32	70.80	58.42	54.31	55.74	16.37	31.25	21.43	—
Llama3.2-90B (Team, 2024)	83.47	60.34	63.28	55.91	49.84	52.36	14.85	25.99	19.54	—
Text-Image Detection Methods										
HAMMER (Shao et al., 2023)	—	—	—	71.23	—	—	48.53	40.86	—	—
FKA-Owl (Liu et al., 2024)	—	—	—	72.08	—	—	—	—	—	—
AMD (Zhang et al., 2025)	—	—	—	65.34	—	—	35.91	37.64	—	—
Text Detection Methods										
Bert (Devlin et al., 2019)	61.45	—	—	—	—	51.37	—	35.91	—	—
LUKE (Yamada et al., 2020)	63.72	—	—	—	—	50.95	—	38.28	—	—
Image Detection Methods										
D^3 (Yang et al., 2025b)	—	88.46	—	—	—	—	—	—	—	—
Fake-VLM (Wen et al., 2025)	—	90.39	—	—	—	—	—	—	—	—
Video Detection Methods										
FakeSV-VLM (Wang et al., 2025a)	—	—	98.81	—	—	—	—	—	—	—
Video-R1 (Feng et al., 2025)	—	—	96.44	—	—	—	—	—	—	—
Segmentation Methods										
SAM3 (Carion et al., 2025)	—	—	—	—	—	—	—	—	—	30.13
OmniVL-Guard (Shen et al., 2026)	96.20	93.12	98.58	75.52	—	—	54.26	63.78	59.22	—
OmniVL-Guard Pro (Ours)	**97.38**	**94.67**	**99.03**	**78.91**	**81.27**	**86.45**	**58.72**	**66.94**	**64.37**	**55.90**
Δ (vs. Best)	+1.18	+1.55	+0.22	+3.39	+26.96	+30.71	+4.46	+3.16	+5.15	+25.77

Table 2: Performance comparison on out-of-domain benchmarks.

		MLLMs		Image-Text Methods			Text Methods		Image Methods		Video-Text Methods		Seg.	
Modality	**Dataset**	Qwen3VL	Llama3.2	HAMMER	FKA-Owl	AMD	Bert	LUKE	D^3	Fake-VLM	FakeSV-VLM	Video-R1	SAM3	**Pro (Ours)**
Text	ISOT	88.74	81.91	—	—	—	33.95	35.78	—	—	—	—	—	**94.12**
Image	CASIA2.0	60.88	58.17	—	—	—	—	—	49.20	51.22	—	—	—	**67.48**
Text-Image	MMFakeBench	57.40	54.14	61.83	62.32	59.11	—	—	—	—	—	—	—	**81.74**
Text-Video	FakeTT	59.84	56.28	—	—	—	—	—	—	—	54.26	52.18	—	**68.06**
Segmentation	IMD	—	—	—	—	—	—	—	—	—	—	—	16.22	**42.93**

capabilities; otherwise, process-level signals would be noisy when the model still fails to perform reliable detection, grounding, or segmentation.

4.1 Stage I: Outcome-Driven RL with ARSPO

We first perform SFT on Qwen3VL-8B using $\mathrm{FSTR}_{\mathrm{sft}}$, obtaining π_{sft}. Only model-generated reasoning, tool calls, and final-answer tokens contribute to the loss; tool-returned observations are masked out. The full formulation is given in Appendix D.5.

Subsequently, we split $\mathrm{FSTR}_{\mathrm{rl}}$ into two parts, $\mathcal{D}_{\mathrm{rl}}^1$ and $\mathcal{D}_{\mathrm{rl}}^2$, where $\mathcal{D}_{\mathrm{rl}}^1$ is used for outcome-level RL, and $\mathcal{D}_{\mathrm{rl}}^2$ is reserved for subsequent Checker-guided process-supervised optimization.

For each task question x in $\mathcal{D}_{\mathrm{rl}}^1$, the current policy π_θ performs inference in the real tool environment and produces a complete response$y = (\tau, \hat{a})$. Then we define the total reward as

$$R(y, x) = R_{\mathrm{ans}}(y, x) + R_{\mathrm{fmt}}(y) + R_{\mathrm{rep}}(y), \tag{2}$$

where R_{ans}, R_{fmt}, and R_{rep} denote the answer reward, format reward, and repetition penalty, respectively. R_{ans} is computed by accuracy for classification tasks, and by overlap-based metrics such as IoU or Dice for grounding and segmentation tasks. Details are provided in Appendix D.1.

We adopt ARSPO (Shen et al., 2026) to optimize this reward, restricting token-level updates to model-generated tokens. The objective and its token-level definitions are provided in Appendix D.5. This stage updates π_{sft} to π_{ans}, the initialization policy for Checker-guided process optimization.

4.2 Stage II: Checker Construction via Rollout Distillation

After obtaining π_{ans}, we construct a Checker to evaluate whether the complete tool reasoning trajectory produced by the Detector provides sufficient and consistent support for its final prediction, thereby avoiding the pseudo-success pattern where the "answer is correct but the process is distorted." For each sample, the Checker takes the question x, the trajectory τ, and the predicted answer $\hat{a}$ as input, and outputs fine-grained evaluation together with an overall score s.

We define the scoring space as three discrete levels: $s \in \{0,\ 0.5,\ 1\}$. Specifically, $s = 1$ indicates that the entire trajectory is internally consistent, the tool usage is reasonable, and the acquired key evidence is sufficient to support the final prediction; $s = 0.5$ indicates that the trajectory is locally reasonable but still suffers from issues such as insufficient evidence or incomplete intermediate support; and $s = 0$ indicates that the trajectory lacks effective support overall, the tool usage and intermediate reasoning fail to form a self-consistent evidence chain, or the final prediction is clearly inconsistent with the reasoning process. Details are provided in Appendix F.1.

We adopt a discrete three-level score rather than a continuous dense score because the Checker is intended to provide process supervision signals with clear boundaries and stable judgments. In contrast, continuous scores often lack fine-grained discriminative criteria; for instance, the model cannot reliably provide a distinguishable boundary between 0.95 and 0.93, which would introduce additional noise.

For each question x in $\mathcal{D}_{\text{rl}}^2$, we use π_{ans} to perform rollout in the real tool environment, obtaining a complete response $y = (\tau, \hat{a})$. Based on these rollout responses, we further construct supervised samples for the Checker.

First, we perform preliminary filtering on the raw responses produced by π_{ans} rollout, removing samples with abnormal formats, incomplete trajectories, or parsing failures. We then construct an MLLM pool $\mathcal{M} = \{\text{GPT5.4},\ \text{Gemini3.1},\ \text{Seed2.0}\}$, and randomly select two MLLMs from it. Each selected MLLM is required to assess the trajectory according to the scoring criteria defined above, outputting both the fine-grained evaluation rationale and the discrete score s. Samples for which the two scores disagree are removed to complete cross-verification. Next, the remaining MLLM verifies the above evaluations and scores, filtering out samples with ambiguous evidence or unclear scoring boundaries. All samples that pass the filtering process jointly constitute the Checker training set $\mathcal{D}_{\text{checker}}$. The construction details are provided in Appendix F.2.

We then concatenate the fine-grained evaluation and the discrete score into the target output sequence, and perform standard autoregressive SFT on Qwen3VL-8B using $\mathcal{D}_{\text{checker}}$, obtaining the process discrimination model π_{checker}.

4.3 Stage III: Checker-Guided Process RL with ARSPO

After obtaining the outcome-level policy π_{ans} and the process discrimination model π_{checker}, we further conduct Checker-guided process-supervised RL, encouraging the model to generate more coherent, trustworthy, and concise tool reasoning trajectories while maintaining answer correctness.

For each task question x in $\mathcal{D}_{\text{rl}}^2$, the current policy performs inference in the real tool environment and produces $y = (\tau, \hat{a})$. We first determine whether the response is correct based on the final answer. For classification tasks, correctness means that the final prediction matches the annotation; for grounding and segmentation tasks, correctness means that the corresponding evaluation metric is higher than 0.7. Only correct responses are further evaluated by π_{checker} to obtain a process score $s_{\text{proc}} \in \{0, 0.5, 1\}$, while incorrect responses are not subjected to Checker-based process penalization, since they have already been penalized by the answer reward.

To avoid unconstrained tool invocation during inference, we further design a tool-use efficiency reward. Specifically, among all correct responses to the same question, the response with the fewest tool calls receives $R_{\text{toolmin}}(y) = 0.5$, while the others receive $R_{\text{toolmin}}(y) = 0$.

Based on the above design, we define the total reward as

$$R(y, x) = \begin{cases} s_{\text{proc}}(y) \times \big(R_{\text{ans}}(y, x) + R_{\text{fmt}}(y) + R_{\text{toolmin}}(y) + R_{\text{rep}}(y)\big), & \text{if correct,} \\ R_{\text{ans}}(y, x) + R_{\text{rep}}(y) + R_{\text{fmt}}(y), & \text{otherwise.} \end{cases} \tag{3}$$

Table 3: Four-component ablation with shared trajectory-generated SFT data and tool environment. Bin.Cls.: five-modality mean accuracy (Table 1); Δ AVG: six-metric mean gain over SFT (points).

Stage I		Stage II	Stage III	Classification		Localization			Seg.	
SFT	ARSPO	Checker Train	Checker RL	Bin. Cls.	RealFact	Image	Text	Video	Dice	Δ AVG
✓	✗	✗	✗	60.74	70.82	28.30	43.48	19.64	32.35	–
✓	✓	✗	✗	88.48	81.36	52.41	63.92	50.35	48.64	+21.64
✓	✓	✗	✓	84.27	77.84	46.83	60.71	40.08	43.27	+16.28
✓	✓	✓	✓	**90.25**	**86.45**	**58.72**	**66.94**	**64.37**	**55.90**	**+27.88**

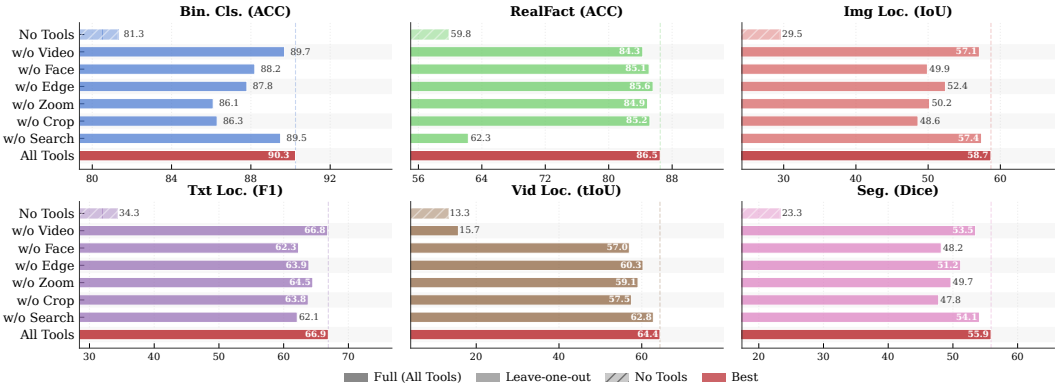


Figure 3: Tool ablation study. Each panel shows the performance of one metric across different tool configurations. The dashed line indicates the All Tools baseline.

where R_{ans}, R_{toolmin}, R_{rep}, and R_{fmt} denote the answer reward, tool-use efficiency reward, repetition penalty, and format reward, respectively.

We still adopt ARSPO to update the policy using this reward, with optimization restricted to model-generated tokens. The full objective is provided in Appendix D.6. This yields a policy jointly optimized for answer correctness, process consistency, and tool-use efficiency.

During RL rollout and evaluation, each generated tool call is executed online, and its observation is fed back before the next step; see Appendix E.

5 Experiments

Implementation details are provided in Appendix D.

Benchmarks and Baselines. We evaluate on both In-Domain and Out-of-Domain (OOD) benchmarks. The In-Domain setting uses the test split from Sec. 3.1, while the OOD setting is designed to assess zero-shot generalization. The OOD suite includes ISOT (Ahmed et al., 2017) for text, CASIA2.0 (Yadav & Vishwakarma, 2024) for image, MMFakeBench (Liu et al., 2025) for image-text, FakeTT (Bu et al., 2024) for video-text, and IMD (Chen et al., 2025) for segmentation. For baselines, general MLLMs (e.g., Qwen3VL-235B) are evaluated in a zero-shot manner. In contrast, domain-specific methods are trained and then directly evaluated on OOD benchmarks without further fine-tuning, in order to assess cross-domain robustness.

5.1 Main Results

In-Domain Comparison. As shown in Table 1, OmniVL-Guard Pro achieves the best in-domain performance across detection, localization, real-time verification, and segmentation. While maintaining strong binary classification across all modalities, it delivers larger gains on fine-grained tasks, improving RealFact verification and segmentation by 30.71% and 25.77% over the prior results, validating the benefits of tool-augmented evidence acquisition and Checker-guided process supervision.

Beyond Table 1, same-tool zero-shot GPT5.4 ReAct scores 80.59 on RealFact versus Pro's 86.45; full same-tool and specialist comparisons are in Appendices I.1 and I.2.

Out-of-Domain Zero-Shot Comparison. As shown in Table 2, OmniVL-Guard Pro achieves the strongest OOD zero-shot performance across unseen benchmarks, outperforming both general MLLMs and domain-specific models. It attains 81.74% and 68.06% accuracy on Text-Image and Text-Video tasks, respectively, and reaches 42.93% Dice on IMD segmentation. These results demonstrate that self-evolving tool reasoning and CGARL improve robustness to unseen multimodal forgeries and generalize beyond classification to fine-grained segmentation. A comparison with the original OmniVL-Guard appears in Appendix I.3.

5.2 Trajectory Generation and Data Quality

We further investigate the impact of the self-evolution iteration rounds N on model performance (Appendix G) and the necessity of the Weakly-Hinted Hard Trajectory Synthesis for covering long-tail hard samples (Appendix H). Open-source teachers improve over Direct GT but trail the standard pool; 99.6% of 500 audited training trajectories show no detected answer-conditioning traces (Appendices C.5 and C.6).

5.3 CGARL Ablation and Process Validation

Ablation of Training Components. All ablation variants use the same training data. As shown in Table 3, SFT alone provides only limited tool-use capability, while adding ARSPO brings a strong average gain of 21.64%, confirming the importance of outcome-level RL. Directly using Qwen3VL-8B as the intermediate process supervisor without training a dedicated Checker is less effective, yielding only 16.28% average improvement, which suggests that generic MLLM feedback can introduce noisy process signals. With the trained Checker and Checker-guided RL, the full model achieves the best performance across all tasks and the largest average gain of 27.88%, validating the necessity of task-adapted process supervision. Jointly removing Stages II and III reduces performance across all six metrics, further supporting the effectiveness of CGARL.

Human audits support process improvement beyond Checker reward: pseudo-success falls from 25.4% to 7.2% (500 trajectories/model), with 96.2% Checker–human agreement (Appendix F.3). Three reduced-scale runs yield 80.46 ± 0.47 Bin.Cls. (Appendix D.8).

5.4 Tool Contributions

Ablation of Tool Modules. Figure 3 shows that the full tool set consistently achieves the best overall performance, while removing all tools causes large drops across all tasks, confirming the necessity of external tool interaction. In particular, search is crucial for RealFact verification, where removing it reduces accuracy from 86.5% to 62.3%. For video localization, removing the video tool leads to the largest degradation, reducing tIoU from 64.4% to 15.7%. These results indicate that different tools provide complementary evidence for different forensic subtasks.

6 Conclusion

We present OmniVL-Guard Pro, a tool-augmented agent that extends unified vision-language forensics from closed-world prediction to open-world evidence-driven reasoning. Through Tree-Structured Self-Evolving Trajectory Generation and Checker-Guided Agentic Reinforcement Learning, our model learns to actively acquire external clues, perform fine-grained inspection, and form evidence-consistent judgments. OmniVL-Guard Pro achieves state-of-the-art performance across all tasks, with strong zero-shot generalization.

AI USE STATEMENT

Generative AI models are used in the research pipeline described in this paper. GPT5.4, Gemini3.1, and Seed2.0 are used for tool-trajectory exploration, guidance, quality assessment, weakly-hinted synthesis, and refinement (Section 3 and Appendix C), and for generating and cross-verifying the scores and rationales used to train the Checker (Section 4.2 and Appendix F). The open-source teacher ablation uses GLM-4.6V-106B, Qwen3VL-235B, and InternVL3.5-241B (Appendix C.5). The RealFact construction pipeline uses MLLMs for claim and evidence processing and manipulation generation; GPT-5.4 and Qwen3VL-235B are also used for leakage, parametric-memory, and evidence-grounded filtering (Appendix B). The trajectory quality checks, cross-model verification procedures, and human verification of a sampled subset of RealFact are described in Appendices C.2, F, and B.3. An AI assistant was additionally used to help polish the manuscript. The authors are responsible for the final content of the paper, including AI-assisted text, claims, and artifacts.

REFERENCES

Sahar Abdelnabi, Rakibul Hasan, and Mario Fritz. Open-domain, content-based, multi-modal fact-checking of out-of-context images via online resources. In *CVPR*, pp. 14920–14929, 2022.

Hadeer Ahmed, Issa Traoré, and Sherif Saad. Detection of online fake news using n-gram analysis and machine learning techniques. In *ISDDC*, pp. 127–138, 2017.

Tobias Braun, Mark Rothermel, Marcus Rohrbach, and Anna Rohrbach. DEFAME: Dynamic evidence-based FAct-checking with multimodal experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 5383–5417. PMLR, 2025. URL https://proceedings.mlr.press/v267/braun25b.html.

Yuyan Bu, Qiang Sheng, Juan Cao, Peng Qi, Danding Wang, and Jintao Li. Fakingrecipe: Detecting fake news on short video platforms from the perspective of creative process. In *ACM MM*, pp. 1351–1360, 2024.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. *CoRR*, 2025.

Haoxing Chen, Yan Hong, Zizheng Huang, Zhuoer Xu, Zhangxuan Gu, Yaohui Li, Jun Lan, Huijia Zhu, Jianfu Zhang, Weiqiang Wang, and Huaxiong Li. Demamba: Ai-generated video detection on million-scale genvideo benchmark. *CoRR*, 2024.

Yuwei Chen, Ming-Ching Chang, Matthias Kirchner, Zhenfei Zhang, Xin Li, Arslan Basharat, and Anthony Hoogs. A semantically impactful image manipulation dataset: Characterizing image manipulations using semantic significance. In *WACV*, pp. 7659–7668, 2025.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, 2025.

Jiankang Deng, Jia Guo, Xiang An, Zheng Zhu, and Stefanos Zafeiriou. Masked face recognition challenge: The insightface track report. In *ICCV Vision Workshops*, pp. 1437–1444, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Benyou Wang, and Xiangyu Yue. Video-r1: Reinforcing video reasoning in mllms. *CoRR*, 2025.

Chang Gao, Chujie Zheng, Xionghui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *CoRR*, 2025.

Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, Haoqi Fan, Jian Wang, Jianyu Jiang, Jiawei Wang, Jingji Chen, Jingjia Huang, Kang Lei, Liping Yuan, Lishu Luo, Pengfei Liu, Qinghao Ye, Rui Qian, Shen Yan, Shixiong Zhao, Shuai Peng, Shuangye Li, Sihang Yuan, Sijin Wu, Tianheng Cheng, Weiwei Liu, Wenqian Wang, Xianhan Zeng, Xiao Liu, Xiaobo Qin, Xiaohan Ding, Xiaojun Xiao, Xiaoying Zhang, Xuanwei Zhang, Xuehan Xiong, Yanghua Peng, Yangrui Chen, Yanwei Li, Yanxu Hu, Yi Lin, Yiyuan Hu, Yiyuan Zhang, Youbin Wu, Yu Li, Yudong Liu, Yue Ling, Yujia Qin, Zanbo Wang, Zhiwu He, Aoxue Zhang, Bairen Yi, Bencheng Liao, Can Huang, Can Zhang, Chaorui Deng, Chaoyi Deng, Cheng Lin, Cheng Yuan, Chenggang Li, Chenhui Gou, Chenwei Lou, Chengzhi Wei, Chundian Liu, Chunyuan Li, Deyao Zhu, Donghong Zhong, Feng Li, Feng Zhang, Gang Wu, Guodong Li, Guohong Xiao, Haibin Lin, Haihua Yang, Haoming Wang, Heng Ji, Hongxiang Hao, Hui Shen, Huixia Li, Jiahao Li, Jialong Wu, Jianhua Zhu, Jianpeng Jiao, Jiashi Feng, Jiaze Chen, Jianhui Duan, Jihao Liu, Jin Zeng, Jingqun Tang, Jingyu Sun, Joya Chen, Jun Long, Junda Feng, Junfeng Zhan, Junjie Fang, Junting Lu, Kai Hua, Kai Liu, Kai Shen, Kaiyuan Zhang, and Ke Shen. Seed1.5-vl technical report. *CoRR*, 2025.

Kilichbek Haydarov, Aashiq Muhamed, Xiaoqian Shen, Jovana Lazarevic, Ivan Skorokhodov, Chamuditha Jayanga Galappaththige, and Mohamed Elhoseiny. Adversarial text to continuous image generation. In *CVPR*, pp. 6316–6326, 2024.

Yinan He, Bei Gan, Siyu Chen, Yichun Zhou, Guojun Yin, Luchuan Song, Lu Sheng, Jing Shao, and Ziwei Liu. Forgerynet: A versatile benchmark for comprehensive forgery analysis. In *CVPR*, pp. 4360–4369, 2021.

Shan Jia, Mingzhen Huang, Zhou Zhou, Yan Ju, Jialing Cai, and Siwei Lyu. Autosplice: A text-prompt manipulated image dataset for media forensics. In *CVPR*, pp. 893–903, 2023.

Yupeng Li, Haorui He, Jin Bai, and Dacheng Wen. MCFEND: A multi-source benchmark dataset for chinese fake news detection. In *WWW*, pp. 4018–4027, 2024a.

Zhengqi Li, Richard Tucker, Noah Snavely, and Aleksander Holynski. Generative image dynamics. In *CVPR*, pp. 24142–24153, 2024b.

Xuannan Liu, Peipei Li, Huaibo Huang, Zekun Li, Xing Cui, Jiahao Liang, Lixiong Qin, Weihong Deng, and Zhaofeng He. Fka-owl: Advancing multimodal fake news detection through knowledge-augmented lvlms. In *ACM MM*, pp. 10154–10163. ACM, 2024.

Xuannan Liu, Zekun Li, Pei-Pei Li, Huaibo Huang, Shuhan Xia, Xing Cui, Linzhi Huang, Weihong Deng, and Zhaofeng He. Mmfakebench: A mixed-source multimodal misinformation detection benchmark for lvlms. In *ICLR*, 2025.

Grace Luo, Trevor Darrell, and Anna Rohrbach. Newsclippings: Automatic generation of out-of-context multimodal media. In *EMNLP*, pp. 6801–6817, 2021.

Peng Qi, Yuyan Bu, Juan Cao, Wei Ji, Ruihao Shui, Junbin Xiao, Danding Wang, and Tat-Seng Chua. Fakesv: A multimodal benchmark with rich social context for fake news detection on short video platforms. In *AAAI*, pp. 14444–14452, 2023.

Rui Shao, Tianxing Wu, and Ziwei Liu. Detecting and grounding multi-modal media manipulation. In *CVPR*, pp. 6904–6913. IEEE, 2023.

Jinjie Shen, Yaxiong Wang, Lechao Cheng, Nan Pu, and Zhun Zhong. Beyond artificial misalignment: Detecting and grounding semantic-coordinated multimodal manipulations. In *ACM MM*, 2025.

Jinjie Shen, Jingjing Wu, Yaxiong Wang, Lechao Cheng, Shengeng Tang, Tianrui Hui, Nan Pu, and Zhun Zhong. Omnivl-guard: Towards unified vision-language forgery detection and grounding via balanced RL. *CoRR*, 2026.

Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big Data*, 8(3):171–188, 2020.

Xiufeng Song, Xiao Guo, Jiache Zhang, Qirui Li, Lei Bai, Xiaoming Liu, Guangtao Zhai, and Xiaohong Liu. On learning multi-modal forgery representation for diffusion generated video detection. In *NeurIPS*, 2024.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, 2025.

Llama Team. The llama 3 herd of models. *CoRR*, 2024.

Junxi Wang, Yaxiong Wang†, Lechao Cheng, and Zhun Zhong†. Fakesv-vlm: Taming vlm for detecting fake short-video news via progressive mixture-of-experts adapter. In *EMNLP*, 2025a.

Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. Opensdi: Spotting diffusion-generated images in the open world. In *CVPR*, pp. 4291–4301, 2025b.

Siwei Wen, Junyan Ye, Peilin Feng, Hengrui Kang, Zichen Wen, Yize Chen, Jiang Wu, Wenjun Wu, Conghui He, and Weijia Li. Spot the fake: Large multimodal model-based synthetic image detection with artifact explanation. In *NeurIPS*, 2025.

Ankit Yadav and Dinesh Kumar Vishwakarma. Towards effective image forensics via A novel computationally efficient framework and A new image splice dataset. *CoRR*, 2024.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. LUKE: deep contextualized entity representations with entity-aware self-attention. In *EMNLP*, pp. 6442–6454. Association for Computational Linguistics, 2020.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jian Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025a.

Yongqi Yang, Zhihao Qian, Ye Zhu, Olga Russakovsky, and Yu Wu. Dˆ3: Scaling up deepfake detection by learning from discrepancy. In *CVPR*, pp. 23850–23859. Computer Vision Foundation / IEEE, 2025b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Junyan Ye, Baichuan Zhou, Zilong Huang, Junan Zhang, Tianyi Bai, Hengrui Kang, Jun He, Honglin Lin, Zihao Wang, Tong Wu, Zhizheng Wu, Yiping Chen, Dahua Lin, Conghui He, and Weijia Li. LOKI: A comprehensive synthetic data detection benchmark using large multimodal models. In *ICLR*, 2025.

Yuchen Zhang, Yaxiong Wang, Yujiao Wu, Lianwei Wu, and Li Zhu. The coherence trap: When mllm-crafted narratives exploit manipulated visual contexts. In *CVPR*, 2025.

Justin Zhao, Timothy Wang, Wael Abid, Geoffrey Angus, Arnav Garg, Jeffery Kinnison, Alex Sherstinsky, Piero Molino, Travis Addair, and Devvret Rishi. Lora land: 310 fine-tuned llms that rival gpt-4, A technical report. *CoRR*, 2024.

APPENDIX

A. Related Work 14

B. Real-time Event Benchmark 14

B.1 Construction Process 14

B.2 Prompt Templates 15

B.3 Human Verification 16

B.4 RealFact Benchmark Examples 17

C. Details of Tool Trajectory Seed Guidance 17

C.1 Guider Judgment in Seed Trajectory Construction 18

C.2 Quality Checking Process 18

C.3 Guider in Weakly-Hinted Hard Trajectory Synthesis 19

C.4 Refiner for Hint Trace Elimination 20

C.5 Teacher-Pool Dependence 21

C.6 Audit of Answer-Conditioned Reasoning 21

D. Implementation Details 21

D.1 Reward Function Configuration 21

D.2 Advantage Weighting Function 23

D.3 Hyperparameter Settings in ARSPO 23

D.4 Source Datasets 23

D.5 SFT and Outcome-Driven Optimization 23

D.6 Checker-Guided Process Optimization 24

D.7 Evaluation Metrics and Numerical Aggregation 24

D.8 Smaller-Scale Seed Sensitivity 24

E. Inference Protocol for Tool Interaction 25

E.1 Action and Stopping Semantics 25

E.2 Retrieval Protocol 25

F. Details of Checker Construction 25

F.1 Scoring Criteria for Process Quality 26

F.2 MLLM Prompts for Checker Construction 27

F.3 Human Evaluation of Process Quality 28

F.4 Timing of Checker Supervision 28

G. Analysis of Self-Evolution Iteration Rounds 29

H. Ablation Study on Weakly-Hinted Hard Trajectory Synthesis 29

I. Additional Baseline Comparisons 30

I.1 Same-Tool ReAct Baselines 30

I.2 Task-Specific and Single-Retrieval Controls 31

I.3 OOD Comparison with the Original OmniVL-Guard 31

J. Limitations 32

J.1 Computational Cost and Open-Source Commitment 32

J.2 Safeguards Against Misuse 32

K. Case Study 33

A Related Work

Forgery Detection and Grounding. With the rapid evolution of generative techniques, detection methodologies have progressed from unimodal paradigms—targeting specific modalities such as images, text, or videos—to bimodal synergies (e.g., image–text, video–text). More recently, OmniVL-Guard establishes a unified framework that simultaneously handles detection and grounding across image, video, and text modalities within a single model. Nevertheless, all these methods operate under a closed-world paradigm that implicitly assumes verification can be completed by the model itself, leaving them vulnerable to manipulations requiring external evidence or fine-grained perceptual scrutiny.

Tool-Augmented Language Agents. The integration of tool use into large language models has given rise to a new class of agents that interact with external environments to accomplish complex tasks. Foundational frameworks such as ReAct and Toolformer demonstrate that LLMs can effectively interleave reasoning with tool invocation. This paradigm has been extended to multimodal settings, where agents leverage specialized tools for visual question answering, image editing, and embodied reasoning. Despite these advances, the application of tool-augmented agents to forensics remains largely unexplored. Forgery detection, grounding, and segmentation present unique challenges: the agent must decide not only when and which tools to invoke, but also how to integrate heterogeneous evidence from diverse sources into a coherent and verifiable judgment.

OmniVL-Guard Pro. Our work extends OmniVL-Guard from a closed-world unified forensic model to an open-world evidence-driven agent. Unlike prior methods that rely solely on parametric knowledge, OmniVL-Guard Pro actively retrieves real-time evidence through web search, performs fine-grained local inspection via cropping and zooming tools, and produces pixel-level manipulation masks using SAM3. To train this agent, we propose Tree-Structured Self-Evolving Trajectory Generation, which addresses the efficiency–bias dilemma inherent in constructing tool-reasoning data: direct MLLM exploration yields low-quality trajectories, while injecting ground-truth cues introduces hindsight bias. Furthermore, we introduce Checker-Guided Agentic Reinforcement Learning (CGARL) to provide process-level supervision that penalizes the pseudo-success pattern where the answer is accidentally correct but the reasoning process is distorted, a failure mode that answer-level rewards alone cannot detect.

B Real-time Event Benchmark

B.1 Construction Process

To evaluate whether models possess real-time event verification capabilities beyond parametric memory, we construct a real-time event benchmark **RealFact** for open-world fact verification. The benchmark covers the period from January 1, 2025 to March 31, 2026, containing 6,208 positive samples and 6,208 negative samples across diverse domains, including politics, economy, technology, society, sports, and more. The construction process is detailed as follows:

▷ *Data Collection:* We first collect candidate real-time events from publicly accessible information sources to build an initial event pool. The sources include mainstream news media, official announcements, as well as public webpages such as sports announcements and policy release pages. Each candidate event is required to satisfy three criteria simultaneously: first, the event time must be explicit and fall within the target time window; second, publicly accessible external textual evidence must be available and cross-supported by multiple independent sources; third, the event must contain verifiable key elements, such as persons, organizations, locations, time, actions, or outcomes. Based on these criteria, we clean and deduplicate the raw candidate events, removing samples with ambiguous timestamps, single-source evidence, or insufficient verifiable support. The retained events constitute the base event set of the benchmark, covering politics, economy, technology, society, sports, disasters, international news, and other domains.

▷ *Evidence-grounded Sample Construction:* After obtaining the base event set, we employ an MLLM to extract key information from each event, forming structured information that includes the event time, subject, location, key action, which is subsequently verified by another MLLM. Based on the structured information, we construct benchmark samples. For positive samples, the

textual claim is consistent with the canonical event representation and can be directly supported by external evidence. For negative samples, we construct deceptive claims around real events, rather than using fabricated statements that are obviously incorrect or semantically incoherent. Specifically, these negative samples are created by modifying the event time, replacing the event subject, altering location information, changing the key action or outcome, or incorrectly stitching together multiple real events.

▷ *Annotation and Verification:* To ensure the quality and difficulty of the benchmark, we further conduct multi-stage filtering on the constructed samples. First, we use GPT-5.4 to check whether samples contain explicit cues that directly leak the answer, and remove samples with obvious hinting information. Next, we use Qwen3VL-235B, whose knowledge base is up to early 2025, to reason over all samples without providing external evidence, filtering out simple samples that can be judged solely from parametric memory. Then, we provide Qwen3VL-235B with the corresponding external evidence and perform inference again, further removing ambiguous samples for which a clear conclusion still cannot be obtained even with evidence. Only samples that pass all the above checks are included in the final benchmark. Finally, we conduct human verification on a randomly sampled subset to validate the quality of the automated construction and filtering pipeline.

B.2 Prompt Templates

This section provides the detailed prompt templates used in the benchmark construction process.

Prompt 1: Key Information Extraction

System Instruction: You are an expert information extraction assistant. Your task is to extract structured information from news articles about real-world events.
Your Task: Extract the following information from the provided news article:

1. *Event Time:* The specific date or time when the event occurred.
2. *Subject:* The person, organization, or entity involved in the event.
3. *Location:* The place where the event took place.
4. *Key Action:* The main action or event that occurred.
5. *Outcome:* The result or consequence of the event.

Input Data: News Article: {article_text}
Output Format: Return strictly in JSON format: { “time”: “...”, “subject”: “...”, “location”: “...”, “action”: “...”, “outcome”: “...” }

Prompt 2: Information Verification

System Instruction: You are an expert fact-checking assistant. Your task is to verify whether extracted information accurately reflects the original news article.
Your Task: Verify each field of the extracted information against the original article:

1. Is the event time correctly extracted?
2. Is the subject correctly identified?
3. Is the location correctly identified?
4. Is the key action correctly described?
5. Is the outcome correctly stated?

Input Data: News Article: {article_text} **Extracted Information:** {extracted_json}
Output Format: For each field, respond with “Correct” or “Incorrect” and provide a brief explanation if incorrect.

Prompt 3: Leakage Check (GPT-5.4)

System Instruction: You are an expert quality assurance assistant. Your task is to check whether benchmark samples contain explicit cues that directly leak the answer.
Your Task: Analyze the claim and evidence to detect any leakage:

1. Does the claim contain obvious factual errors that can be detected without verification?
2. Does the claim contain self-contradictory information?
3. Does the claim contain explicit hints about its truthfulness?

Input Data: Claim: {claim_text} **Evidence:** {evidence_text}
Output Format: Respond with "No Leakage" if the claim does not contain obvious cues, or "Leakage Detected" with explanation.

Prompt 4: Parametric Memory Reasoning (Qwen3VL-235B)

System Instruction: You are a knowledgeable assistant. Based solely on your internal knowledge (without any external evidence), determine whether the given claim is true or false.
Your Task: Analyze the claim and provide your judgment:

1. Consider the factual accuracy of the claim.
2. Consider the temporal consistency (does the time match known events?).
3. Consider the plausibility of the described event.

Input Data: Claim: {claim_text}
Output Format: Respond with:

- "True" if you believe the claim is factually correct.
- "False" if you believe the claim is factually incorrect.
- "Uncertain" if you cannot determine the truthfulness from your knowledge.

Prompt 5: Evidence-Grounded Reasoning (Qwen3VL-235B)

System Instruction: You are an expert fact-verification assistant. Based on the provided external evidence, determine whether the given claim is true or false.
Your Task: Analyze the claim against the evidence:

1. Does the evidence support the claim?
2. Is there any contradiction between the claim and the evidence?
3. Is the evidence sufficient to make a definitive judgment?

Input Data: Claim: {claim_text} **Evidence:** {evidence_text}
Output Format: Respond with:

- "Supported" if the evidence confirms the claim is true.
- "Refuted" if the evidence contradicts the claim.
- "Insufficient" if the evidence is not enough to make a judgment.

B.3 Human Verification

To validate the quality of the automated construction and filtering pipeline, we conduct human verification on a randomly sampled subset of the benchmark. This subsection details the verification process, criteria, and results.

▷ *Verification Criteria:*

1. **Claim Authenticity:** Does the claim accurately reflect the original event? For positive samples, the claim should be factually consistent with the event. For negative samples, the introduced manipulation should be realistic and non-trivial.
2. **Evidence Sufficiency:** Is the provided evidence sufficient to verify or refute the claim? The evidence should be directly relevant and conclusive.
3. **Temporal Consistency:** Is the event timestamp accurate and consistent with the claim? The temporal information should be verifiable through the evidence.
4. **Manipulation Subtlety:** For negative samples, is the manipulation subtle enough to challenge models? Obvious or trivial manipulations are flagged for removal.

▷ *Verification Results:* 98.6% of the sampled instances are accepted, indicating high quality of the automated pipeline.

These results demonstrate that our automated construction pipeline produces high-quality benchmark samples, with the vast majority meeting human standards for factuality, evidence sufficiency, and manipulation subtlety.

B.4 REALFACT BENCHMARK EXAMPLES

Figure 4 presents two representative examples from the RealFact benchmark, illustrating a fake sample and a real sample.



Figure 4: Representative examples from the RealFact benchmark. (a) A fake sample with manipulated content. (b) A real sample with verified factual content.

C DETAILS OF TOOL TRAJECTORY SEED GUIDANCE

This section provides additional details for the Tree-Structured Self-Evolving Tool Trajectory Generation process described in Sec. 3. Specifically, we elaborate on four key components: (1) the *Guider* judgment criteria used during seed trajectory construction (Sec. C.1); (2) the two-stage quality checking process for filtering high-quality seed trajectories (Sec. C.2); (3) the *Guider*'s dual role in weakly-hinted hard trajectory synthesis, where it both judges tool steps and provides weak hints to guide the Explorer (Sec. C.3); and (4) the *Refiner* that eliminates explicit hint traces to produce autonomous-looking trajectories (Sec. C.4).

C.1 GUIDER JUDGMENT IN SEED TRAJECTORY CONSTRUCTION

During the tree-structured seed trajectory construction, the *Guider* plays a critical role in evaluating each candidate tool step. At each step t, given the current history $\{(r_i, u_i, o_i)\}_{i=1}^{t-1}$ and the final answer $a^\star$, the *Guider* assesses whether the candidate step (r_t, u_t, o_t) is valid and helpful for solving the task. The *Guider* judges the step based on whether the selected tool is appropriate for the current reasoning state, whether the tool parameters are reasonable, whether the returned observation provides useful information, and whether the step advances the reasoning toward the correct answer. If the step is judged invalid, the branch is immediately pruned; otherwise, the step is retained for further expansion.

The detailed prompt used for the *Guider* judgment is provided below.

Prompt for Guider Judgment

System Instruction: You are an expert Guider assisting an Explorer agent in constructing high-quality tool-use trajectories for vision-language forensics tasks. You have access to the ground truth answer $a^\star$.
Your Task: Evaluate whether the candidate tool step is valid and helpful for solving the task. Consider the following criteria:

1. **Tool Relevance:** Is the selected tool appropriate for the current task and reasoning state?
2. **Parameter Quality:** Are the tool parameters (e.g., search query, crop box, zoom region) specific and reasonable?
3. **Observation Utility:** Does the returned observation provide useful, non-redundant information?
4. **Progress Toward Answer:** Does this step bring the reasoning closer to the ground truth answer $a^\star$?

Input Data: Task Question: {question} **Ground Truth Answer:** $a^\star$ **Current History:** $\{(r_i, u_i, o_i)\}_{i=1}^{t-1}$ **Candidate Step:** (r_t, u_t, o_t)
Output Format: Return strictly in JSON format: { "judgment": "Valid" or "Invalid", "reason": "Brief explanation of the judgment." }

C.2 QUALITY CHECKING PROCESS

After the *Explorer* completes the search and outputs the final prediction $\hat{a}$, we apply a two-stage quality checking process to ensure the trajectory meets the required standards.

▷ *Stage 1: Outcome-Based Filtering:* We first filter trajectories based on the correctness of the final prediction $\hat{a}$:

- **Classification Tasks:** Trajectories with incorrect final answers are removed. Only trajectories where $\hat{a}$ matches the ground truth label $a^\star$ are retained.
- **Image Grounding Tasks:** We compute the Intersection over Union (IoU) between the predicted bounding box and the ground truth annotation. Only trajectories with IoU ≥ 0.7 are retained.
- **Text Grounding Tasks:** We compute the F1 score between the predicted text span and the ground truth annotation. Only trajectories with F1 ≥ 0.7 are retained.
- **Video Grounding Tasks:** We compute the temporal Intersection over Union (tIoU) between the predicted temporal segment and the ground truth annotation. Only trajectories with tIoU ≥ 0.7 are retained.
- **Segmentation Tasks:** We compute the Dice score between the predicted mask and the ground truth mask. Only trajectories with Dice ≥ 0.7 are retained.
- **Real-time Event Verification:** Trajectories with incorrect verification results (True/False) are removed.

▷ *Stage 2: Trajectory Quality Assessment:* For trajectories that pass Stage 1, we further assess the overall quality using the following criteria:

1. **Trajectory Coherence:** The overall trajectory should be logically coherent, with each step building upon previous steps without abrupt jumps or contradictions.
2. **Evidence Support:** The intermediate reasoning steps and tool usage should collectively support the final conclusion. The evidence gathered should directly verify or refute the claim, reveal manipulations, or localize relevant regions.
3. **No Ineffective Steps:** The trajectory should not contain redundant tool invocations, empty observations, or reasoning steps that do not contribute to the final conclusion.
4. **Completeness:** The trajectory should contain sufficient evidence to support the prediction, rather than arriving at the correct answer through lucky guessing.

The detailed prompt used for trajectory quality assessment is provided below.

Prompt for Trajectory Quality Assessment

System Instruction: You are an expert quality assessor for tool-use trajectories in vision-language forensics. Your task is to evaluate whether a trajectory meets the required quality standards.
Your Task: Assess the trajectory based on the following criteria:

1. **Coherence:** Is the trajectory logically coherent without abrupt jumps or contradictions?
2. **Evidence Support:** Do the reasoning steps and tool usage collectively support the final conclusion?
3. **Efficiency:** Does the trajectory contain ineffective or redundant steps?
4. **Completeness:** Is there sufficient evidence to support the prediction?

Input Data: Task Question: {question} **Ground Truth Answer:** $a^\star$ **Trajectory:** $\tau = \{(r_t, u_t, o_t)\}_{t=1}^{T} \cup \{\hat{a}\}$ **Final Prediction:** $\hat{a}$
Output Format: Return strictly in JSON format: { "quality": "Pass" or "Fail", "reason": "Brief explanation of the quality assessment." }

Only trajectories that pass both stages of the quality checking process are included in the seed set $\mathcal{D}_s^0$. This rigorous filtering ensures that the seed trajectories used for SFT are of high quality, providing a strong foundation for subsequent self-evolution and RL training.

C.3 Guider in Weakly-Hinted Hard Trajectory Synthesis

In the Weakly-Hinted Hard Trajectory Synthesis stage (Sec. 3.4), the *Guider* plays a dual role compared to the Seed Guidance stage. Beyond judging whether the current tool step is valid, the *Guider* also provides *weak hints*—necessary parameters or local clues related to the current tool invocation—to narrow down the search space for the *Explorer*. For example, in image scenarios, the *Guider* provides a rough candidate region to assist cropping or zooming; in text scenarios, it may suggest keywords or entity names for search queries. These weak hints help the *Explorer* overcome the difficulty of exploring hard samples, while keeping the hints sufficiently vague to avoid directly leaking the answer.

The detailed prompt used for the *Guider* with weak hints is provided below.

Prompt for Guider with Weak Hints

System Instruction: You are an expert Guider assisting an Explorer agent in constructing tool-use trajectories for hard samples in vision-language forensics. You have access to the ground truth answer $a^\star$. Your role is to evaluate tool steps and provide weak hints to guide the Explorer.
Your Task: For each candidate tool step, perform two actions:

1. **Judgment:** Evaluate whether the tool step is valid based on tool relevance, parameter quality, observation utility, and progress toward the answer.
2. **Weak Hint:** If the step is invalid or suboptimal, provide a weak hint to guide the Explorer. The hint should be:
   - Sufficient to narrow down the search space (e.g., a rough region for cropping, keywords for search).
   - Vague enough to avoid directly revealing the answer $a^\star$.
   - Focused on local clues rather than global conclusions.

Input Data: Task Question: {question} **Ground Truth Answer:** $a^\star$ **Current History:** $\{(r_i, u_i, o_i)\}_{i=1}^{t-1}$ **Candidate Step:** (r_t, u_t, o_t)
Output Format: Return strictly in JSON format: { "judgment": "Valid" or "Invalid", "hint": "Weak hint for the Explorer (empty if valid)", "reason": "Brief explanation of the judgment." }

C.4 Refiner for Hint Trace Elimination

After obtaining the weakly-hinted candidate trajectories, we employ a *Refiner* to eliminate explicit traces introduced by the weak hints. The *Refiner* is a randomly selected model from the MLLM expert pool $\mathcal{M}$ (distinct from the Explorer and Guider), which rewrites the reasoning process r_t at each step of the trajectory. The goal is to remove answer-aware hints from the reasoning, preventing the model from learning hint-dependent reasoning patterns that do not generalize to real inference scenarios where no hints are available.

Specifically, the *Refiner* rewrites each reasoning step to:

1. Remove explicit references to the hint (e.g., "the Guider suggested region X" $\rightarrow$ "based on visual analysis, region X appears suspicious").
2. Rephrase the reasoning to sound as if it were generated autonomously by the Explorer.
3. Preserve the factual content and logical flow of the original reasoning.

The detailed prompt used for the *Refiner* is provided below.

Prompt for Refiner (Hint Trace Elimination)

System Instruction: You are an expert Refiner responsible for rewriting reasoning processes in tool-use trajectories. Your task is to eliminate explicit traces of weak hints that were provided by a Guider, so that the final trajectory appears as if it were generated autonomously without any external guidance.
Your Task: For each reasoning step r_t in the trajectory, rewrite it to:

1. Remove any explicit references to hints (e.g., "the Guider suggested...", "based on the hint...").
2. Rephrase the reasoning to sound natural and autonomous, as if the Explorer discovered the information through its own analysis.
3. Preserve the factual content, logical flow, and tool invocation decisions.
4. Ensure the rewritten reasoning is consistent with the tool invocation u_t and the returned observation o_t.

Input Data: Task Question: {question} **Original Trajectory:** $\tau = \{(r_t, u_t, o_t)\}_{t=1}^{T} \cup \{\hat{a}\}$
Weak Hints Used: {hints provided by the Guider at each step}
Output Format: Return the rewritten trajectory with hint traces eliminated: $\tau' = \{(r'_t, u_t, o_t)\}_{t=1}^{T} \cup \{\hat{a}\}$ where r'_t is the rewritten reasoning process for step t.

All trajectories that pass the Refiner and the subsequent quality checking process (same as in Sec. C.2) are included in the hard-sample set $\mathcal{D}_{sh}$.

C.5 Teacher-Pool Dependence

To assess whether the pipeline retains benefits without the standard closed-source teachers, we compare Direct GT, open-source teachers with Self-Evolving, and the standard setting on a 7k subset. The replacement teacher pool consists of GLM-4.6V-106B, Qwen3VL-235B, and InternVL3.5-241B. Table 4 uses the six-metric protocol defined in Appendix D.7.

Table 4: Teacher-pool comparison on the 7k subset.

Setting	Bin.Cls.	RealFact	Img.Loc.	Text.Loc.	Vid.Loc.	Seg.
Direct GT	71.55	64.59	34.52	48.53	32.98	30.36
Open-source teachers						
+ Self-Evolving	76.30	69.94	41.73	53.76	36.92	34.28
Standard	**80.82**	**74.56**	**44.67**	**58.71**	**45.25**	**40.38**

Table 4 shows that the open-source setting improves all six metrics over Direct GT: Bin.Cls. increases from 71.55 to 76.30 and RealFact from 64.59 to 69.94. These improvements demonstrate that the alternative teacher configuration provides useful gains without the standard closed-source pool. Nevertheless, the standard setting reaches 80.82 and 74.56 on these metrics and remains better on every task, indicating that teacher quality still affects performance. The gains over Direct GT across both teacher pools further support the effectiveness of the trajectory-generation and self-evolution mechanisms in our pipeline.

C.6 Audit of Answer-Conditioned Reasoning

To assess residual answer conditioning after ground-truth-aware trajectory construction, we randomly sample 500 FSTR_{sft} trajectories and manually inspect them for answer-conditioned reasoning. During construction, the seed Guider uses ground truth for judging and pruning rather than exposing it to the Explorer; hard-sample synthesis uses weak hints, followed by Refiner rewriting and quality checking (Appendices C.1, C.3, and C.4).

No answer-conditioned traces are detected in 99.6% of the sample. This result provides empirical support for the effectiveness of our trajectory-generation pipeline in limiting answer-conditioned reasoning.

D Implementation Details

We train our model on 32 NVIDIA H200 GPUs (141 GB) using Qwen3VL-8B as the backbone, and adopt AdamW throughout all training stages. In Stage I of CGARL, the SFT phase uses a batch size of 32, a learning rate of 6×10^{-5}, and 3 epochs, while the RL phase uses a batch size of 64 (i.e., 64 sampled trajectories per step), a learning rate of 1.2×10^{-6}, 2 epochs, and a KL penalty of 0.01. For the three fine-grained tasks—image tampering localization, text tampering localization, video temporal tampering localization and segmentation—the parameter α in the reward mapping function is consistently set to 3. The data scale ratio between $\mathcal{D}_{\text{rl}}^{1}$ and $\mathcal{D}_{\text{rl}}^{2}$ is approximately $7:3$.

D.1 Reward Function Configuration

In the RL phase, the total reward R for a generated response y given question x and ground truth $a^{\star}$ is composed of three distinct components: the task-specific performance score (R_{task}), the structural format reward (R_{fmt}), and a repetition penalty (R_{rep}). The final reward is calculated as:

$$R(y, x) = R_{\text{task}}(y, x) + R_{\text{fmt}}(y) + R_{\text{rep}}(y) \tag{4}$$

1. Task-Specific Performance Reward (R_{task})
We employ distinct metrics for different forensic tasks. To encourage the model to strive for high-precision grounding rather than mediocre overlap, we apply an exponential mapping function to continuous metrics (IoU, F1, tIoU).

- **Binary Forgery Classification:** For the binary classification task, we use a simple discrete reward based on correctness:

$$R_{\text{task}}^{\text{cls}} = \begin{cases} 1, & \text{if } \hat{a} = a^\star, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

- **Tampering Localization (Image, Text, Video):** For fine-grained localization tasks, we compute a raw metric $m \in [0, 1]$ specific to the modality: Intersection over Union (IoU) for image bounding boxes, F1-score for text indices, and Temporal IoU (tIoU) for video segments. The raw metric is mapped to a normalized reward using an exponential function with a scaling factor $\alpha = 3$:

$$R_{\text{task}}^{\text{loc}}(m) = \frac{e^{\alpha \cdot m} - 1}{e^{\alpha} - 1} \tag{6}$$

  This convex mapping ($\alpha = 3$) amplifies the reward signal for high-quality responses (e.g., $m > 0.7$) while suppressing rewards for low-quality overlaps.

- **Forgery Segmentation:** For the segmentation task, we compute a composite score that combines box-level IoU and point prompt consistency. Specifically, we first convert the predicted bounding box into a binary mask M_{pred} (1 inside the box, 0 outside) of the same size as the ground truth mask M_{gt}, and compute the Box IoU:

$$\text{IoU}_{\text{box}} = \frac{|M_{\text{pred}} \cap M_{\text{gt}}|}{|M_{\text{pred}} \cup M_{\text{gt}}|} \tag{7}$$

  We then evaluate point prompt consistency: for each predicted point, we check whether its label is consistent with the ground truth mask (i.e., positive points should fall inside M_{gt} and negative points should fall outside). The point consistency score is computed as:

$$\text{score}_{\text{point}} = \frac{\text{number of consistent points}}{\text{total number of points}} \tag{8}$$

  The final segmentation score is a weighted combination:

$$m_{\text{seg}} = 0.7 \times \text{IoU}_{\text{box}} + 0.3 \times \text{score}_{\text{point}} \tag{9}$$

  clipped to $[0, 1]$. This composite score is then mapped through the same exponential function with $\alpha = 3$:

$$R_{\text{task}}^{\text{seg}}(m_{\text{seg}}) = \frac{e^{\alpha \cdot m_{\text{seg}}} - 1}{e^{\alpha} - 1} \tag{10}$$

2. Format Compliance Reward (R_{fmt})
To enforce the tool trajectory structure, we employ a strict regular expression check. The model receives a positive reward if and only if the output strictly follows the required format:

$$R_{\text{fmt}} = \begin{cases} 0.2 & \text{if format is valid} \\ 0.0 & \text{otherwise} \end{cases} \tag{11}$$

3. Repetition Penalty (R_{rep})
To prevent degenerate generation loops, we implement an N-gram repetition penalty. We calculate the ratio of unique N-grams to total N-grams in the generated text. With a set hyperparameters of $N = 3$ and a maximum penalty coefficient $\lambda_{\text{pen}} = -1.0$, the penalty is defined as:

$$R_{\text{rep}} = \lambda_{\text{pen}} \times \left(1 - \frac{|S_{\text{unique}}^N|}{N_{\text{total}}}\right) \tag{12}$$

where $|S_{\text{unique}}^N|$ is the count of unique 3-grams and N_{total} is the total number of 3-grams. This term introduces a negative reward proportional to the redundancy of the text.

D.2 Advantage Weighting Function

The token-level weighting function $f_{i,t}(\cdot)$ in Eq. (16) follows the SAPO design (Gao et al., 2025). Specifically, for each response y_i with normalized advantage $\hat{A}_{i,k}$, the weighting function is defined as:

$$f_{i,t}^{\text{sapo}}(r_{i,t}(\theta)) = \frac{4}{\tau_i}\sigma(\tau_i(r_{i,t}(\theta)-1)), \quad \tau_i = \begin{cases} \tau_{\text{pos}}, & \widehat{A}_{i,k} > 0, \\ \tau_{\text{neg}}, & \widehat{A}_{i,k} \le 0, \end{cases} \tag{13}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|x,y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x,y_{i,<t})}$ is the probability ratio, τ_{neg} and τ_{pos} denote the temperature parameters for positive and negative tokens, respectively, and $\sigma(x) = \frac{1}{1+e^{-x}}$ represents the sigmoid function.

D.3 Hyperparameter settings in ARSPO

Table 5: Hyperparameter settings for the Dynamic Coefficient Adjustment Algorithm in ARSPO.

Symbol	Description	Value
Schedule & Initialization		
T_{warm}	Warm-up phase duration (steps)	800
T	Coefficient update interval (steps)	100
Adjustment Dynamics		
α_{boost}	Coefficient boost factor	1.1
α_{decay}	Coefficient decay factor	0.9
Thresholds		
ϵ_{mom}	Momentum protection threshold	0.02
ϵ_{rescue}	Regression rescue threshold	0.10
τ_{high}	High-performance threshold	0.60

D.4 Source Datasets

We first collect raw data from a series of existing datasets to cover the major modalities and task types involved in unified vision-language forensics. Specifically, for the uni-modal text domain, we adopt FakeNewsCorpus (Shu et al., 2020) and MCFEND (Li et al., 2024a); for the single-image modality, we select FakeClue (Wen et al., 2025), LOKI (Ye et al., 2025), ForgeryNet (He et al., 2021), OpenSDI (Wang et al., 2025b), and AutoSplice (Jia et al., 2023); for the video modality, we use GenVideo (Chen et al., 2024), DVF (Song et al., 2024), and ForgeryNet (He et al., 2021); for the image–text modality, we adopt SAMM (Shen et al., 2025), MDSM (Zhang et al., 2025), DGM^4 (Shao et al., 2023), and NewsCLIPpings (Luo et al., 2021); and for the text–video modality, we select FakeSV (Qi et al., 2023). These datasets jointly cover diverse tasks related to forgery detection, manipulation grounding, and semantic segmentation. For the real-time event verification task, we use RealFact constructed in the Appendix B.

D.5 SFT and Outcome-Driven Optimization

We first perform SFT on Qwen3VL-8B using FSTR_{sft}, obtaining the initial policy π_{sft}. For a sample with a tool trajectory, we write it as

$$\tau = \{(r_t, u_t, o_t)\}_{t=1}^{T} \cup \{\hat{a}\}, \tag{14}$$

where the definitions of r_t, u_t, o_t, and $\hat{a}$ are consistent with Eq. 1. Since o_t is returned by the environment rather than generated by the model, we do not compute loss over these tokens during SFT. Instead, the language modeling loss is only applied to model-generated tokens corresponding to r_t, u_t and $\hat{a}$.

Let the complete token sequence of τ, including all T reasoning-tool interaction steps and the final answer, be $y = (y_1, \ldots, y_{|y|})$, and define a token-level mask $m_i \in \{0, 1\}$, where positions corresponding to tool-returned observations o_t take $m_i = 0$, and all other model-generated positions take $m_i = 1$. The SFT loss is therefore defined as

$$\mathcal{L}_{\text{sft}} = - \sum_{(x,\tau)\in \text{FSTR}_{\text{sft}}} \frac{1}{\sum_{i=1}^{|y|} m_i} \times \sum_{i=1}^{|y|} m_i \log \pi_\theta(y_i \mid x, y_{<i}). \tag{15}$$

In this way, the model learns how to generate reasonable tool invocations, parameter specifications, intermediate reasoning, and final answers, rather than fitting the contents returned by the tools.

We adopt ARSPO (Shen et al., 2026) to update the policy, whose optimization objective is formulated as follows:

$$\mathcal{J}_{\text{arspo}}(\theta) = \sum_{k=1}^{K} \frac{|\mathcal{D}^1_{\text{rl},k}|}{|\mathcal{D}^1_{\text{rl}}|} \mathop{\mathbb{E}}_{\substack{x\sim\mathcal{D}^1_{\text{rl},k},\\ \{y_i\}\sim\pi_{\theta_{\text{old}}}}} \left[\frac{l_{k,s}}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} f_{i,t}\big(r_{i,t}(\theta)\big) \hat{A}_{i,k} \right], \tag{16}$$

where K denotes the number of task categories, $\mathcal{D}^1_{\text{rl},k} \subset \mathcal{D}^1_{\text{rl}}$ denotes the data subset corresponding to the k-th task, and G denotes the number of sampled responses for each question. y_i denotes the i-th complete response, and $|y_i|$ is its total token length. $\hat{A}_{i,k} = \frac{A_{i,k}-\mu}{\sigma}$ is the normalized advantage, with μ and σ being the mean and standard deviation of the rewards $\{A_{i,k}\}_{i=1}^{G}$. $f_{i,t}(\cdot)$ is the token-level weighting function in the form of SAPO (Gao et al., 2025) (Detailed in Appendix D.2), and $l_{k,s}$ denotes the dynamic coefficient for the k-th task at training step s. $r_{i,t} = \frac{\pi_\theta(y_{i,t}|q,y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|q,y_{i,<t})}$ is the probability ratio.

Consistent with the SFT stage, the token-level optimization in Eq. (16) is only applied to model-generated tokens. After this stage, the model is updated from π_{sft} to π_{ans}, which serves as the initialization policy for subsequent Checker-guided process-supervised optimization.

D.6 Checker-Guided Process Optimization

We still adopt ARSPO to update the policy, with the optimization objective defined as follows:

$$\begin{aligned} \mathcal{J}^{\text{proc}}_{\text{arspo}}(\theta) = \sum_{k=1}^{K} \frac{|\mathcal{D}^2_{\text{rl},k}|}{|\mathcal{D}^2_{\text{rl}}|} \mathbb{E}_{\substack{x\sim\mathcal{D}^2_{\text{rl},k},\\ \{y_i\}\sim\pi_{\theta_{\text{old}}}}} \\ \times \left[\frac{l_{k,s}}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} f_{i,t}\big(r_{i,t}(\theta)\big) \hat{A}_{i,k} \right], \end{aligned} \tag{17}$$

The token-level optimization in Eq. (17) is only applied to model-generated tokens. After this stage, the model is jointly optimized in terms of answer correctness, process consistency, and tool-use efficiency, ultimately yielding the process-supervision-enhanced policy.

D.7 Evaluation Metrics and Numerical Aggregation

In six-metric tables, Bin.Cls. denotes the mean classification accuracy across text, image, video, text-image, and text-video, with RealFact reported separately. Image, text, and video localization use IoU, F1, and tIoU, respectively; segmentation uses Dice. Scores are on a 0–100 scale, and absolute differences are metric points, not relative percentage improvements. Unavailable or inapplicable results are marked "—".

D.8 Smaller-Scale Seed Sensitivity

To assess the stability of CGARL across random seeds, we run the three-stage procedure using 7k SFT samples and 8k RL samples with seeds 42, 2026, and 1111. This is approximately one-tenth of the full data scale. Table 6 reports the individual runs and their arithmetic mean with sample standard deviation (denominator $3 - 1$).

Across the six metrics, the sample standard deviations range from 0.13 to 0.93 points, with Bin.Cls. at 80.46 ± 0.47.

Table 6: Three-seed evaluation at reduced training scale.

Seed	Bin.Cls.	RealFact	Img.Loc.	Text.Loc.	Vid.Loc.	Seg.
42	80.82	74.56	44.67	58.71	45.25	40.38
2026	79.93	75.44	43.76	56.98	44.96	40.57
1111	80.64	75.03	44.38	58.45	45.24	40.31
Mean $\pm$ SD	80.46 ± 0.47	75.01 ± 0.44	44.27 ± 0.46	58.05 ± 0.93	45.15 ± 0.16	40.42 ± 0.13

E Inference Protocol for Tool Interaction

At inference time, OmniVL-Guard Pro is executed as an interactive tool-augmented agent rather than a one-shot trajectory generator. Given an input question x, the policy first generates a reasoning segment r_t and, when external clues or fine-grained inspection is needed, emits a structured tool invocation u_t following the predefined action format. The environment parses u_t, executes the corresponding tool in $\mathcal{T}$, and returns the observation o_t, such as retrieved evidence, a cropped or zoomed image region, detected face boxes, extracted video frames, or a SAM3 segmentation mask. The observation o_t is then appended to the context, and the policy continues generation conditioned on the updated interaction history $\{(r_i, u_i, o_i)\}_{i=1}^{t}$. This reasoning–action–observation loop is repeated until the policy emits a final answer $\hat{a}$ or reaches the maximum interaction budget.

Formally, for each test sample x, inference proceeds as

$$r_t, u_t \sim \pi_\theta(\cdot \mid x, \{(r_i, u_i, o_i)\}_{i=1}^{t-1}), \qquad o_t = \mathrm{Env}(u_t),$$

where $\mathrm{Env}(\cdot)$ denotes the tool environment described in Sec. 2. The final prediction is produced only after the model has incorporated the observations returned by its previous tool calls:

$$\hat{a} \sim \pi_\theta(\cdot \mid x, \{(r_i, u_i, o_i)\}_{i=1}^{T}).$$

Therefore, the predicted trajectory is not merely a textual explanation generated after the fact; it records the actual sequence of tool invocations and observations used during inference.

We use the same tool interfaces, action schema, and observation format as in trajectory construction and RL training. To avoid unbounded exploration, inference is constrained by a maximum number of tool calls $T_{\max}$. If a generated tool invocation is malformed or refers to an unavailable tool, the environment returns an error observation and the model is allowed to continue, subject to the same interaction budget. Please refer to Appendix K for examples.

E.1 Action and Stopping Semantics

To specify how tree-structured exploration maps to tool execution, we define each action u_t as a call to one tool. The tree structure arises by branching over candidate actions from the same state; a sample can therefore yield multiple candidate trajectories. The policy stops when it emits a final answer after gathering sufficient evidence, or when it reaches the interaction limit $T_{\max}$. Example interaction traces are in Appendix K.

E.2 Retrieval Protocol

To acquire current evidence for RealFact verification, we use Tavily with dynamic queries at evaluation time rather than a fixed evidence corpus. We use repeated inference per sample to reduce sensitivity to individual retrieval outcomes. The trained retrieval-only Qwen3VL-8B baseline makes its prediction from a single retrieval; the full agent can iteratively collect and integrate evidence. Their task-accuracy comparison is reported in Appendix I.2.

F Details of Checker Construction

This section provides additional details for the Checker construction process described in Sec. 4.2. Specifically, we elaborate on the scoring criteria for the discrete three-level score $s \in \{0, 0.5, 1\}$ and the prompts used for MLLM-based evaluation and verification.

F.1 Scoring Criteria for Process Quality

The Checker assigns a discrete score $s \in \{0, 0.5, 1\}$ to evaluate the quality of the tool reasoning trajectory. The detailed criteria for each score level are as follows:

Score $s = 1$: Fully Consistent and Sufficient. A trajectory receives the highest score when it satisfies all of the following conditions:

- **Internal Consistency:** The reasoning steps form a logically coherent chain without contradictions or abrupt jumps. Each step builds upon previous observations and maintains a clear narrative flow.
- **Reasonable Tool Usage:** The selected tools are appropriate for the current task and reasoning state. Tool parameters (e.g., search queries, crop boxes, zoom regions) are specific and well-justified. The model demonstrates understanding of when and how to invoke each tool.
- **Sufficient Evidence:** The acquired observations collectively provide sufficient evidence to support the final prediction. The evidence chain is complete, with no critical gaps or missing links that would undermine the conclusion.
- **Prediction-Process Alignment:** The final prediction $\hat{a}$ is clearly and directly supported by the accumulated evidence and reasoning. There is no contradiction between the reasoning process and the final answer.

Score $s = 0.5$: Partially Consistent with Limitations. A trajectory receives a moderate score when it exhibits local correctness but suffers from one or more of the following issues:

- **Insufficient Evidence:** While the reasoning is locally sound, the trajectory fails to gather enough evidence to conclusively support the prediction. Some tool calls may return uninformative observations, or the model may prematurely conclude without exploring all relevant aspects.
- **Incomplete Intermediate Support:** The trajectory contains reasonable steps, but some intermediate conclusions are not fully supported by the preceding evidence. There may be minor logical gaps or assumptions that are not adequately validated.
- **Redundant but Non-Detracting Steps:** The trajectory includes some redundant tool calls or reasoning steps that do not contribute to the final conclusion, but do not actively mislead the reasoning process.
- **Weak Prediction Alignment:** The final prediction is plausible given the evidence, but the connection between the reasoning and the conclusion is not explicitly articulated or relies on implicit assumptions.

Score $s = 0$: Inconsistent or Insufficient. A trajectory receives the lowest score when it exhibits one or more of the following critical issues:

- **Logical Inconsistency:** The reasoning chain contains contradictions, where later steps contradict earlier observations or conclusions. The trajectory lacks a coherent narrative structure.
- **Inappropriate Tool Usage:** The model selects inappropriate tools for the task, provides unreasonable parameters, or fails to interpret tool outputs correctly. The tool calls do not contribute meaningfully to the reasoning process.
- **Missing Evidence Chain:** The trajectory lacks a self-consistent evidence chain. Critical evidence is missing, and the final prediction is not supported by the available observations.
- **Prediction-Process Mismatch:** The final prediction $\hat{a}$ is clearly inconsistent with the reasoning process. The model may arrive at the correct answer through lucky guessing or flawed reasoning, which constitutes a pseudo-success pattern.
- **Format Violations:** The trajectory exhibits severe format issues, such as missing tool invocations, incomplete responses, or malformed output structures.

F.2 MLLM Prompts for Checker Construction

We provide the prompts used for the MLLM-based evaluation and verification in the Checker construction process.

Prompt for Generating Score and Rationale. The following prompt is used by the two evaluation MLLMs to assess the trajectory and generate both the fine-grained evaluation rationale and the discrete score s:

Prompt for Evaluation MLLM (Score and Rationale Generation)

System Instruction: You are an expert evaluator for tool-use trajectories in vision-language forensics. Your task is to assess the quality of a reasoning trajectory that uses external tools to verify the authenticity of content. You must provide a detailed evaluation rationale and assign a discrete score based on the criteria below.
Scoring Criteria:
Score 1 (Fully Consistent and Sufficient): The trajectory satisfies all of the following: (a) Internal Consistency—reasoning steps form a logically coherent chain without contradictions or abrupt jumps; (b) Reasonable Tool Usage—selected tools are appropriate for the task and reasoning state, and parameters are specific and well-justified; (c) Sufficient Evidence—acquired observations collectively provide sufficient evidence to support the final prediction, with no critical gaps; (d) Prediction-Process Alignment—the final prediction $\hat{a}$ is clearly and directly supported by the accumulated evidence and reasoning.
Score 0.5 (Partially Consistent with Limitations): The trajectory exhibits local correctness but suffers from one or more of the following: (a) Insufficient Evidence—reasoning is locally sound but fails to gather enough evidence to conclusively support the prediction; (b) Incomplete Intermediate Support—some intermediate conclusions are not fully supported by the preceding evidence; (c) Redundant but Non-Detracting Steps—some redundant tool calls or reasoning steps that do not contribute to the final conclusion but do not actively mislead; (d) Weak Prediction Alignment—the final prediction is plausible but the connection between reasoning and conclusion is not explicitly articulated.
Score 0 (Inconsistent or Insufficient): The trajectory exhibits one or more critical issues: (a) Logical Inconsistency—reasoning chain contains contradictions where later steps contradict earlier observations or conclusions; (b) Inappropriate Tool Usage—inappropriate tool selection, unreasonable parameters, or incorrect interpretation of tool outputs; (c) Missing Evidence Chain—critical evidence is missing and the final prediction is not supported by the available observations; (d) Prediction-Process Mismatch—the final prediction is clearly inconsistent with the reasoning process (pseudo-success pattern); (e) Format Violations—severe format issues such as missing tool invocations, incomplete responses, or malformed output structures.
Input Data: Task Question: {question} **Ground Truth Answer:** $a^\star$ **Trajectory:** $\tau = \{(r_t, u_t, o_t)\}_{t=1}^{T} \cup \{\hat{a}\}$ **Final Prediction:** $\hat{a}$
Output Format: Return strictly in JSON format: { "evaluation": "Detailed analysis of the trajectory quality. Specifically address: (1) whether the reasoning steps are internally consistent, (2) whether the tool usage is appropriate and well-parameterized, (3) whether the acquired evidence is sufficient to support the conclusion, and (4) whether the final prediction aligns with the reasoning process.", "score": 0, 0.5, or 1 }

Prompt for Verification. After an evaluation MLLM generates its evaluation and score, a third MLLM from the pool $\mathcal{M}$ independently verifies each evaluation. Specifically, for each evaluation-scorpair produced by an evaluation MLLM, the verification MLLM checks whether the evaluation is reasonable and the score is justified. The verification is performed one evaluation at a time, and the verification prompt is as follows:

Prompt for Verification MLLM (Single Evaluation Verification)

System Instruction: You are a quality assurance verifier for tool-use trajectory evaluations. An evaluator has assessed a trajectory and provided an evaluation rationale along with a score. Your task is to verify whether the evaluation is reasonable, accurate, and well-justified.
Verification Criteria:

- Check whether the evaluation accurately reflects the trajectory quality according to the scoring criteria.
- Identify any factual errors in the evaluation (e.g., incorrect claims about tool usage or evidence).
- Assess whether the evidence cited in the evaluation is accurate and relevant to the trajectory.
- Determine if the score is justified given the evaluation rationale.

Input Data: Task Question: {question} **Ground Truth Answer:** $a^\star$ **Trajectory:** $\tau = \{(r_t, u_t, o_t)\}_{t=1}^{T} \cup \{\hat{a}\}$ **Final Prediction:** $\hat{a}$ **Evaluation:** {evaluation from evaluation MLLM} **Score:** {score from evaluation MLLM}
Output Format: Return strictly in JSON format: { "verification": "Detailed analysis of whether the evaluation is reasonable and accurate. Highlight any factual errors or inconsistencies found.", "is_valid": true or false, "reason": "Explanation for the verification decision. If invalid, specify which aspects of the evaluation contain errors." }

All samples that pass the verification process, where both evaluation MLLMs agree on the score and the verification MLLM confirms the validity, are included in the Checker training set $\mathcal{D}_{\text{checker}}$.

F.3 Human Evaluation of Process Quality

To assess CGARL's process validity, we manually evaluate 500 in-domain test trajectories from the Stage-I-only model and 500 from full CGARL. We define pseudo-success as a correct final answer accompanied by an invalid process or evidence that does not support the conclusion. To evaluate the reliability of the process supervisor, we also compare Checker decisions with human trajectory-quality judgments on 500 trajectories. These evaluations measure the policy's failure mode and the supervisor's agreement with human judgments, respectively.

Table 7: Human process evaluation. Sample sizes refer to the trajectories evaluated for each reported result.

Evaluation	Trajectories	Rate (%)
Pseudo-success: Stage I ↓	500	25.4
Pseudo-success: full CGARL ↓	500	7.2
Checker–human agreement ↑	500	96.2

Table 7 shows that full CGARL reduces pseudo-success from 25.4% to 7.2% relative to the Stage-I-only model, an 18.2-percentage-point reduction. Under the human evaluation, this means fewer trajectories produce a correct final answer through an invalid process or evidence that does not support the conclusion. The result supports CGARL's effectiveness in improving reasoning–evidence consistency beyond Stage-I outcome-driven optimization. Separately, the 96.2% Checker–human agreement shows that the learned supervisor's process judgments closely match human assessments, supporting its use as a process-reward signal. Together, the reduction in human-identified pseudo-success and the human alignment of Checker judgments provide complementary evidence for CGARL's process-level effectiveness.

F.4 Timing of Checker Supervision

To train the Checker on rollouts from a tool-capable policy, we construct it after Stage I. During initial trajectory construction, the Guider and quality gates instead use strong MLLMs to judge answer correctness and trajectory consistency. Stage I produces the policy whose rollouts supply the Stage-II

process-supervision data. This ordering aligns Checker construction with the availability of policy rollouts.

G Analysis of Self-Evolution Iteration Rounds

To determine the optimal termination point for the Tree-Structured Self-Evolving Tool Trajectory Generation strategy (Sec. 3.3), we investigated the performance trajectory across iterations n. We hypothesized that the self-evolution process would eventually reach a saturation point where the teacher policy π_n no longer provides superior tool-use signals compared to π_{n-1}, thereby yielding no further quality improvements in the generated dataset $\mathrm{FSTR}_{\mathrm{sft}}^{n}$.

Specifically, we compared the capabilities of the model trained after the fourth iteration against a model extended to a fifth iteration. We constructed a fifth-round dataset $\mathrm{FSTR}_{\mathrm{sft}}^{5}$ using π_4 and combined it with the hard-sample set $\mathcal{D}_{sh}$ to train the full OmniVL-Guard Pro framework via CGARL. We then compared this against our standard setting ($\mathrm{FSTR}_{\mathrm{sft}}^{4} \cup \mathcal{D}_{sh}$).

Table 8: Performance comparison between stopping at Iteration 4 ($\mathrm{FSTR}_{\mathrm{sft}}^{4}$) vs. Iteration 5 ($\mathrm{FSTR}_{\mathrm{sft}}^{5}$). The results indicate that performance plateaus at $n = 4$, justifying the termination of the self-evolution loop to conserve computational resources.

Training Set	**Binary Classification (ACC)**					**Localization & Segmentation**			
	Text	Image	Video	Text-Image	RealFact	Image (IoU)	Text (F1)	Video (tIoU)	Seg. (Dice)
$\mathrm{FSTR}_{\mathrm{sft}}^{4} \cup \mathcal{D}_{sh}$ (Ours)	97.38	94.67	99.03	78.91	86.45	58.72	66.94	64.37	55.90
$\mathrm{FSTR}_{\mathrm{sft}}^{5} \cup \mathcal{D}_{sh}$	97.21	94.89	98.98	79.15	86.34	58.55	67.23	64.19	56.01
Diff (Δ)	*+0.17*	*-0.22*	*+0.05*	*-0.24*	*+0.11*	*+0.17*	*-0.29*	*+0.18*	*-0.11*

As illustrated in Table 8, the performance differences between the fourth and fifth iterations are statistically negligible. For instance, the Image Localization (IoU) shifted marginally from 58.72 to 58.55 ($\Delta = +0.17$), while Text Localization (F1) experienced a slight fluctuation from 66.94 to 67.23 ($\Delta = -0.29$). The results show a mix of minor improvements and degradations across tasks, with no consistent gains. This evidence suggests that the quality of the tool trajectory data generated by π_4 has reached a bottleneck, and further iterations do not yield distinguishable gains in downstream forensic tasks. Consequently, we terminate the self-evolution process at $n = 4$ to strike an optimal balance between model performance and training efficiency.

H Ablation Study on Weakly-Hinted Hard Trajectory Synthesis

To validate the necessity of the **Weakly-Hinted Hard Trajectory Synthesis** stage (Sec. 3.4), we conducted an ablation study by excluding the hard-sample subset $\mathcal{D}_{sh}$ from the SFT cold-start dataset.

Experimental Settings:

- **Full Setting (Ours):** The model is fine-tuned on the complete dataset $\mathrm{FSTR}_{\mathrm{sft}} = \mathcal{D}_{s}^{4} \cup \mathcal{D}_{sh}$, followed by CGARL training.
- **w/o Hard Samples:** The model is fine-tuned only on the self-evolved dataset $\mathcal{D}_{s}^{4}$ (removing the synthesized hard samples $\mathcal{D}_{sh}$), followed by the same CGARL training.

The results are reported in Table 9. We observe that removing the hard samples leads to a consistent performance degradation across all tasks.

Analysis of Results:

- **Impact on Fine-grained Localization and Segmentation:** The performance drop is significant in localization and segmentation tasks. Specifically, Text Localization (F1) drops by **9.63%**, Video Temporal Localization (tIoU) drops by **7.95%**, and Segmentation (Dice) drops by **7.67%**. This empirical evidence suggests that the self-evolution loop alone tends to saturate on easier samples. The failure cases (hard samples) from the iterative process often contain subtle manipulations or complex multimodal inconsistencies that require the

Table 9: Ablation study on the necessity of Weakly-Hinted Hard Trajectory Synthesis. We report the performance comparison between using the full dataset versus removing the synthesized hard samples ($\mathcal{D}_{sh}$). The results show that $\mathcal{D}_{sh}$ is critical for maintaining high performance across both classification and localization tasks.

Method	Binary Classification (ACC)					Localization & Segmentation			
	Text	Image	Video	Text-Image	RealFact	Image (IoU)	Text (F1)	Video (tIoU)	Seg. (Dice)
Ours (Full FSTR_{sft})	**97.38**	**94.67**	**99.03**	**78.91**	**86.45**	**58.72**	**66.94**	**64.37**	**55.90**
w/o Hard Samples ($\mathcal{D}_{sh}$)	93.45	88.72	92.18	71.56	82.23	52.84	57.31	56.42	48.23
Performance Drop (Δ)	-3.93	-5.95	-6.85	-7.35	-4.22	-5.88	-9.63	-7.95	-7.67

specific, high-quality tool-use trajectories provided by the Weakly-Hinted Hard Trajectory Synthesis to resolve.

- **Impact on Binary Classification:** The binary classification accuracy also decreases notably (e.g., **-7.35%** on Text-Image). This indicates that the hard-sample data is indispensable not only for fine-grained grounding and segmentation but also for maintaining robust detection capabilities in complex scenarios.

In conclusion, the Weakly-Hinted Hard Trajectory Synthesis effectively complements the self-evolution pipeline by covering the long-tail distribution of difficult forgery types, preventing the model from overfitting to simple patterns.

I Additional Baseline Comparisons

To evaluate whether OmniVL-Guard Pro's performance advantage persists beyond the original baselines, we compare it with tool-enabled agents, task-specific alternatives, and its forensic predecessor. Results use the in-domain evaluation unless marked OOD; metric definitions are in Appendix D.7.

I.1 Same-Tool ReAct Baselines

To assess whether tool access alone can reproduce OmniVL-Guard Pro's performance, we evaluate six general MLLMs in a zero-shot ReAct loop (Yao et al., 2022) with the same tool environment as OmniVL-Guard Pro. The comparison includes the same-size Qwen3VL-8B, larger open models, and the GPT5.4, Gemini3.1, and Seed2.0 teachers used in data construction. Table 10 also includes the tool-free Qwen3VL-235B and Llama3.2-90B results.

Table 10: Tool-free and same-tool zero-shot baselines. OmniVL-Guard Pro uses task-specific training; the ReAct baselines do not.

Method	Bin.Cls.	RealFact	Img.Loc.	Text.Loc.	Vid.Loc.	Seg.
Tool-free, zero-shot						
Qwen3VL-235B	67.62	55.74	16.37	31.25	21.43	—
Llama3.2-90B	62.57	52.36	14.85	25.99	19.54	—
ReAct with the same tools, zero-shot						
Qwen3VL-8B	59.31	61.28	13.11	22.99	16.32	14.11
Qwen3VL-235B	70.93	71.26	19.39	32.98	19.80	21.97
Llama3.2-90B	62.88	59.92	16.30	26.07	17.56	15.34
GPT5.4	78.56	80.59	24.31	38.97	31.67	28.72
Gemini3.1	75.39	77.92	23.85	37.95	29.58	25.70
Seed2.0	73.34	75.31	23.67	38.03	27.44	27.59
Pro (8B)	**90.25**	**86.45**	**58.72**	**66.94**	**64.37**	**55.90**

Table 10 shows that OmniVL-Guard Pro exceeds all six same-tool ReAct baselines on all six metrics. Against Qwen3VL-8B, Bin.Cls. is 90.25 versus 59.31 and RealFact is 86.45 versus 61.28; GPT5.4 obtains 80.59 on RealFact. Video localization decreases for both larger open models after enabling ReAct.

The complementary stage ablations in Table 3 keep the initial SFT trajectories and tool set fixed: average gains over SFT are 21.64 points after Stage I, 27.88 with full CGARL, and 16.28 with an untrained process supervisor. Together, these results support the added value of task-adapted process supervision rather than generic MLLM feedback.

I.2 Task-Specific and Single-Retrieval Controls

To evaluate whether OmniVL-Guard Pro's gains persist against specialized and simpler task-specific alternatives, we compare with DEFAME (Braun et al., 2025) on RealFact, FakeSV-VLM (Wang et al., 2025a) and Video-R1 (Feng et al., 2025) trained for Text-Video, and a trained Qwen3VL-8B that predicts from one retrieval. The trained and untrained SAM3 (Carion et al., 2025) settings share localization prompts.

Table 11: Task-specific comparisons. The paired SAM3 settings share localization prompts.

Method	RealFact	Text-Video	Seg. (Dice)
DEFAME	80.45	—	—
FakeSV-VLM	—	78.51	—
Video-R1	—	76.93	—
Qwen3VL-8B (single retrieval, trained)	73.85	—	—
SAM3 (no training)	—	—	14.68
SAM3 (trained)	—	—	30.13
Pro	**86.45**	**81.27**	**55.90**

In Table 11, OmniVL-Guard Pro exceeds DEFAME by 6.00 RealFact points (86.45 versus 80.45) and FakeSV-VLM by 2.76 Text-Video points (81.27 versus 78.51); Video-R1 reaches 76.93. The advantage therefore persists beyond the tool-free general MLLMs.

For single-retrieval verification, the trained retrieval-only Qwen3VL-8B scores 73.85 on RealFact, 12.60 points below OmniVL-Guard Pro.

For segmentation, untrained and trained SAM3 obtain 14.68 and 30.13 Dice under the paired localization prompts, versus OmniVL-Guard Pro's 55.90.

I.3 OOD Comparison with the Original OmniVL-Guard

To quantify the cross-domain improvement over the original forensic model, we compare OmniVL-Guard Pro with OmniVL-Guard (Shen et al., 2026) without OOD fine-tuning. Table 12 reports the results on the shared OOD benchmarks and the available IMD segmentation result.

Table 12: OOD comparison with the original OmniVL-Guard. The first four columns report accuracy; IMD reports Dice. No OOD fine-tuning is applied.

Method	ISOT	CASIA2.0	MMFakeBench	FakeTT	IMD
OmniVL-Guard	93.69	63.64	79.38	64.38	—
Pro	**94.12**	**67.48**	**81.74**	**68.06**	**42.93**

OmniVL-Guard Pro improves accuracy on all four shared OOD benchmarks by 0.43 points on ISOT, 3.84 on CASIA2.0, 2.36 on MMFakeBench, and 3.68 on FakeTT. These results show an empirical cross-domain improvement over the predecessor, in addition to the gains over general MLLMs. IMD reaches 42.93 Dice, but the original model has no reported segmentation result.

J LIMITATIONS

J.1 COMPUTATIONAL COST AND OPEN-SOURCE COMMITMENT

Both the construction of the FSTR dataset and the multi-stage training of OmniVL-Guard Pro demand substantial computational resources, including large-scale tool-environment rollouts, multi-iteration self-evolution, and multi-stage RL optimization. Our goal is to establish a foundation model for unified vision-language forensics that supports diverse downstream tasks. To lower the barrier for the community and facilitate future research, we commit to fully open-sourcing the FSTR dataset, the trained OmniVL-Guard Pro checkpoints, and all associated code, enabling researchers to directly fine-tune or evaluate on their own tasks without reproducing the expensive training pipeline.

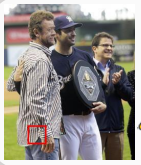
J.2 SAFEGUARDS AGAINST MISUSE

Given that the RealFact benchmark, FSTR dataset, and OmniVL-Guard Pro model could potentially be repurposed for malicious activities such as generating or evading forgery detection, we will implement a strict access control process. All assets will be released exclusively for academic research purposes, and each applicant will be required to provide institutional affiliation and a clear statement of intended use. Access will be granted only after a manual review, and any violation of the terms will result in immediate revocation.

K CASE STUDY

Case1 – Text-Image Forgery Detection

Question: Please determine if the image is **Real**, **AI-generated**, or **Tampered**.



Caption: *Braun poses with Hall of Famer Robin Yount and his 2011 National League MVP award.*

Tampered



OmniVL-Guard Pro:

<think>Okay, let's see. The user provided a news image with a caption saying Braun poses with Hall of Famer Robin Yount and his 2011 National League MVP award. I need to check if this is real, AI-generated, or tampered.

First, I should verify the event. The caption mentions Ryan Braun winning the 2011 NL MVP. I know that Ryan Braun did win the MVP in 2011, so that part is correct. But the image shows him with Robin Yount, who is a Hall of Famer. Wait, Robin Yount is a former Milwaukee Brewers player, and he was inducted into the Hall of Fame in 1999. So the context here might be plausible.

But maybe there's something off. Let me think. The user wants to check if the image is real or tampered. Since it's a photo, possible issues could be deepfake, editing, or incorrect elements.

The tools available include detect_edge_anomaly, which can find possible tampering. But first, since it's a single image (not a video), I don't need to use the video extraction tool. The image index is 0 (since it's the first image provided).

So, maybe run detect_edge_anomaly on the image to see if there are any suspicious regions. If there are anomalies, then it might be tampered. If not, maybe it's real. Also, check faces with detect_faces to see if the people match known individuals.

Wait, the caption mentions Robin Yount. Let me confirm: Robin Yount is a real person, a Hall of Famer. Ryan Braun is also real. The 2011 MVP award is accurate. So the content seems plausible, but the image could still be edited.

Let me proceed step by step. First, check for edge anomalies. Call detect_edge_anomaly on image_index 0. That might highlight areas where the image was altered. If the tool returns some regions, then investigate those further with zoom or crop. If no anomalies, maybe the image is real.

So the first tool call should be detect_edge_anomaly with image_index 0.</think>

<tool_call>{"name": "**detect_edge_anomaly**", "arguments": {"image_index": 0}}</tool_call>

<tool_response>{"status": "success", "images": "<image>", "output": {"suspicious_boxes": [{"score": 2.6052839756011963, "box": [188, 784, 312, 889], "edge_density": 0.294921875, "lap_energy": 73.44140625, "neighbor_gap": 8.88311767578125}, {"score": 2.3950839042663574, "box": [188, 523, 312, 627], "edge_density": 0.3505859375, "lap_energy": 77.76953125, "neighbor_gap": 3.6822509765625}, {"score": 2.363237142562866, "box": [188, 575, 312, 680], "edge_density": 0.3515625, "lap_energy": 76.7265625, "neighbor_gap": 3.54803466796875}, {"score": 2.3256735801696777, "box": [188, 627, 312, 732], "edge_density": 0.35546875, "lap_energy": 74.46484375, "neighbor_gap": 3.46331787109375}, {"score": 2.3091726303100586, "box": [188, 418, 312, 523], "edge_density": 0.3408203125, "lap_energy": 75.4140625, "neighbor_gap": 3.7838134765625}], "global_stats": {"block_size": 32, "stride": 16, "canny_low": 60, "canny_high": 160, "num_windows_y": 18, "num_windows_x": 15, "max_score": 2.6052839756011963, "mean_score": 0.20154379308223724}}}</tool_response>



<think>The edge anomaly detection highlighted several suspicious regions on the left person's shirt (vertical stripes). These anomalies could indicate digital tampering, such as inconsistent edge patterns or blending artifacts. To investigate further, I should zoom into one of these high-score regions to examine the texture and details. The highest-scoring box is [188, 784, 312, 889], which corresponds to the lower part of the striped shirt. Magnifying this area with a scale of 3.0 will help determine if the pattern shows signs of manipulation like unnatural repetition or blurring.</think>

<tool_call>{"name": "**zoom_image**", "arguments": {"image_index": 0, "x1": 188, "y1": 784, "x2": 312, "y2": 889, "scale": 3.0}}</tool_call>



<tool_response>{"status": "success", "images": "<image>", "output": {"box": [188, 784, 312, 889], "scale": 3.0}}</tool_response>

<think>The zoomed-in view of the suspicious region shows highly distorted and inconsistent stripe patterns with unnatural blurring and pixelation. This suggests digital tampering, likely involving AI-based editing to alter the original image. The irregularities in the fabric texture are not typical of natural photography artifacts. Given the edge anomaly results combined with the visual inconsistencies in the magnified region, the image appears to be tampered with.</think>

<answer>**Answer: Tampered**</answer>

Figure 5: Case study 1.

Case2 – Text Tampering Location

Question: Please locate the tampered words in the caption and provide the indices of the altered words as a list (starting from 0).



Caption: *John Swinney expressed that he was in the depths of despair but was doing everything he currently could to support the economy.*

[6, 8, 9, 10, 11]



OmniVL-Guard Pro:

<think>I need to analyze whether any words in the caption are inconsistent with the image content. First, I'll break down the caption into individual words with their indices for reference:

0: John 1: Swinney 2: expressed 3: that 4: he 5: was 6: in 7: the 8: depths 9: of 10: despair 11: but 12: was 13: doing 14: everything 15: he 16: currently 17: could 18: to 19: support 20: the 21: economy

The image shows a man holding documents labeled "SCOTTISH BUDGET," which suggests a formal economic context. The phrase "in the depths of despair" stands out as emotionally intense and potentially inconsistent with the professional tone expected in a budget presentation. I'll start by examining the image for any visual anomalies that might support or contradict the caption's emotional tone. I'll use detect_edge_anomaly to check for manipulated regions, especially around facial features or text elements.</think>

<tool_call>{"name": "**detect_edge_anomaly**", "arguments": {"image_index": 0}}</tool_call>



<tool_response>{"status": "success", "images": "<image>", "output": {"suspicious_boxes": [{"score": 62853.09375, "box": [656, 70, 688, 94], "edge_density": 0.20703125, "lap_energy": 23.193359375, "neighbor_gap": 3.9722900390625}, {"score": 61372.89453125, "box": [656, 82, 688, 106], "edge_density": 0.2021484375, "lap_energy": 25.552734375, "neighbor_gap": 4.49603271484375}, {"score": 57216.84375, "box": [812, 821, 844, 845], "edge_density": 0.1884765625, "lap_energy": 25.056640625, "neighbor_gap": 0.88128662109375}, {"score": 56920.2734375, "box": [781, 845, 812, 868], "edge_density": 0.1875, "lap_energy": 19.703125, "neighbor_gap": 2.0367431640625}, {"score": 55436.93359375, "box": [797, 833, 828, 856], "edge_density": 0.1826171875, "lap_energy": 19.482421875, "neighbor_gap": 1.3216552734375}], "global_stats": {"block_size": 32, "stride": 16, "canny_low": 60, "canny_high": 160, "num_windows_y": 84, "num_windows_x": 63, "max_score": 62853.09375, "mean_score": 4072.0302734375}}</tool_response>

<think>The detect_edge_anomaly tool identified suspicious regions near the top-right (around the forehead/hairline) and bottom-right (on the documents). The highest-scoring anomaly is at [656,70,688,94], which may relate to facial features, and another at [781,845,812,868] on the documents.

The document text "SCOTTISH BUDGET" is clearly visible, confirming the context is a formal economic announcement. The emotional phrase "in the depths of despair" seems out of place here — budget presentations typically involve measured, factual language, not expressions of personal despair. I'll now zoom into the document area to verify if the text has been altered or if there's any visual evidence supporting the caption's tone. I'll use zoom_image on the document region to examine the text more closely.</think>



<tool_call>{"name": "**zoom_image**", "arguments": {"image_index": 0, "x1": 700, "y1": 750, "x2": 900, "y2": 950, "scale": 3.0}}</tool_call>

<tool_response>{"status": "success", "images": "<image>", "output": {"box": [700, 750, 900, 950], "scale": 3.0}}</tool_response>

<think>The zoomed image confirms the document reads "SCOTTISH BUDGET 2024-25," which aligns with John Swinney's role as Scotland's First Minister presenting the budget. This reinforces that the context is professional and economic, not emotional. The phrase "in the depths of despair" (indices 6,8,9,10) is highly inconsistent with this setting. Additionally, the word "but" (index 11) creates a logical contradiction — it's odd to say someone is "in the depths of despair" while simultaneously "doing everything he currently could to support the economy." The key issue remains the caption's tone versus the image's context. The man in the image has a serious but composed expression, not one of despair. The phrase "in the depths of despair but" (indices 6,8,9,10,11) is the most likely tampered section — it's emotionally exaggerated and logically inconsistent with a budget presentation. Words like "expressed" and "that" (indices 2,3) are plausible in this context and likely unchanged. Based on the visual and contextual mismatch, I conclude the tampered words are at indices [6,8,9,10,11].</think>

<answer>**Answer: [6, 8, 9, 10, 11]**</answer>

Figure 6: Case study 2.

Case3 – Image Tampering Location

Question: Please locate the tampered region in the image.



Caption: *Clint Dempsey leads the celebrations after Seattle s US Open Cup final victory.*

[455, 195, 575, 449]

OmniVL-Guard Pro:

<think>The image shows a soccer match with players wearing Germany and another team's jerseys, while the caption references Clint Dempsey and Seattle's US Open Cup final — a clear mismatch. This discrepancy suggests the image may have been altered. I'll start by examining the central area where facial features appear, as that's often a target for manipulation. I'll zoom into that region to inspect for unnatural distortions or inconsistencies.</think>



<tool_call>{"name": "**zoom_image**", "arguments": {"image_index": 0, "x1": 455, "y1": 195, "x2": 575, "y2": 449, "scale": 3.0}}</tool_call>

<tool_response>{"status": "success", "images": "<image>", "output": {"box": [455, 195, 575, 449], "scale": 3.0}}</tool_response>

<think>After zooming in, I notice a facial region that appears unnaturally stretched and misaligned with surrounding elements like the soccer ball and jersey. The contours and proportions don't match typical human anatomy, which raises suspicion of digital tampering. To further investigate, I'll run edge anomaly detection on the full image to see if automated analysis confirms irregularities in this area.</think>

<tool_call>{"name": "**detect_edge_anomaly**", "arguments": {"image_index": 0}}</tool_call>



<tool_response>{"status": "success", "images": "<image>", "output": {"suspicious_boxes": [{"score": 10.478888511657715, "box": [529, 500, 611, 625], "edge_density": 0.259765625, "lap_energy": 80.658203125, "neighbor_gap": 12.66790771484375}, {"score": 9.203973770141602, "box": [489, 438, 570, 562], "edge_density": 0.244140625, "lap_energy": 73.990234375, "neighbor_gap": 9.82305908203125}, {"score": 9.034083366394043, "box": [529, 562, 611, 688], "edge_density": 0.234375, "lap_energy": 70.841796875, "neighbor_gap": 10.41748046875}, {"score": 8.233757972717285, "box": [570, 688, 651, 812], "edge_density": 0.251953125, "lap_energy": 59.744140625, "neighbor_gap": 8.01226806640625}, {"score": 8.169635772705078, "box": [529, 625, 611, 750], "edge_density": 0.232421875, "lap_energy": 63.3515625, "neighbor_gap": 8.4432373046875}], "global_stats": {"block_size": 32, "stride": 16, "canny_low": 60, "canny_high": 160, "num_windows_y": 15, "num_windows_x": 23, "max_score": 10.478888511657715, "mean_score": 1.0289173126220703}}</tool_response>

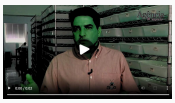
<think>The edge anomaly detection highlights several high-scoring boxes clustered around the region I previously zoomed into. The top-scoring box [529, 500, 611, 625] overlaps significantly with the area showing distorted facial features, reinforcing the presence of manipulation. Additionally, the mismatch between the caption (Clint Dempsey, Seattle, US Open Cup) and the visual content (Germany jersey, World Cup-style ball) further supports that this image has been altered. Based on both visual inspection and anomaly detection, the tampered region is confirmed.</think>

<answer>**Answer: [455, 195, 575, 449].**</answer>

Figure 7: Case study 3.

Case4 – Video Tampering Location

Question: Please identify the specific time segments that have been tampered with.



["0.80s-1.20s"]

OmniVL-Guard Pro:

<think>I need to identify tampered time segments in the video. The video shows a person in a room with shelves. At around 0.8s, there's a noticeable change where the person seems to be holding something different. Let me extract frames from a short segment around this time to analyze further. I'll start by extracting frames from 0.5s to 1.5s to cover the area where the change appears to occur.</think>

<tool_call>{"name": "**extract_video_segment_frame**", "arguments": {"start_sec": 0.5, "end_sec": 1.5}}</tool_call>

<tool_response>{"status": "success", "images": "<image><image><image><image>", "output": {"time_range_sec": [0.5, 1.5], "raw_frame_range": [10, 30], "raw_frame_count_in_range": 21, "sampled_frame_count_for_mllm": 4, "sampled_frame_indices_for_mllm": [10, 17, 23, 30], "sampled_frame_timestamps_sec_for_mllm": [0.5, 0.85, 1.15, 1.5], "sampled_fps_for_mllm": 3.809524, "sampled_image_count": 4}}</tool_response>

<think>The extraction returned 4 frames from 0.5s to 1.5s. The sampled timestamps are 0.5s, 0.85s, 1.15s, and 1.5s. Looking at these frames, the most noticeable change occurs between 0.85s and 1.15s where the object the person is holding changes from a flat item to a different shape. To check for tampering, I should run edge anomaly detection on the frames around this transition. I'll start with the frame at 0.85s (image_index 1) since it's near the middle of the suspected change.</think>

<tool_call>{"name": "**detect_edge_anomaly**", "arguments": {"image_index": 1}}</tool_call>



</tool_response>{"status": "success", "images": "<image>", "output": {"suspicious_boxes": [{"score": 2.794658660888672, "box": [943, 104, 1000, 208], "edge_density": 0.169921875, "lap_energy": 44.541015625, "neighbor_gap": 7.452148914337158}, {"score": 2.701683759689331, "box": [943, 312, 1000, 416], "edge_density": 0.177734375, "lap_energy": 40.744140625, "neighbor_gap": 7.488965034484863}, {"score": 2.5701098442077637, "box": [600, 779, 657, 883], "edge_density": 0.18359375, "lap_energy": 34.578125, "neighbor_gap": 8.0152587890625}, {"score": 2.5143895149230957, "box": [286, 260, 343, 364], "edge_density": 0.1796875, "lap_energy": 43.201171875, "neighbor_gap": 5.494934082031125}, {"score": 2.0997695922851562, "box": [257, 260, 314, 364], "edge_density": 0.212890625, "lap_energy": 36.857421875, "neighbor_gap":2.8558349609375}], "global_stats": {"block_size": 32, "stride": 16, "canny_low": 60, "canny_high": 160, "num_windows_y": 18, "num_windows_x": 34, "max_score": 2.794658660888672, "mean_score": 0.08402005583047867}}</tool_response>

<think>The edge anomaly detection on image_index 1 (0.85s) shows several suspicious boxes, particularly around the logo area and some background elements. However, the logo area might be a natural high-edge region. The most relevant anomaly for tampering would likely be related to the object the person is holding, which changed between frames. Let me check the next frame at 1.15s (image_index 2) to see if there's a similar anomaly in the object region. I'll run edge anomaly detection on image_index 2 to compare.</think>

<tool_call>{"name": "**detect_edge_anomaly**", "arguments": {"image_index": 2}}</tool_call>



<tool_response>{"status": "success", "images": "<image>", "output": {"suspicious_boxes": [{"score": 2.9673092365264893, "box": [943, 312, 1000, 416], "edge_density": 0.19921875, "lap_energy": 41.36328125, "neighbor_gap": 7.39882755279541}, {"score": 2.7297520637512207, "box": [943, 104, 1000, 208], "edge_density": 0.162109375, "lap_energy": 44.736328125, "neighbor_gap": 6.869726657867432}, {"score": 2.5150146484375, "box": [286, 260, 343, 364], "edge_density": 0.1767578125, "lap_energy": 42.673828125, "neighbor_gap": 5.3355712890625}, {"score": 2.356698513031006, "box": [600, 831, 657, 935], "edge_density": 0.197265625, "lap_energy": 36.07421875, "neighbor_gap": 5.0654296875}, {"score": 2.223613739013672, "box": [686, 0, 743, 104], "edge_density": 0.205078125, "lap_energy": 32.783203125, "neighbor_gap": 4.711621284484863}], "global_stats": {"block_size": 32, "stride": 16, "canny_low": 60, "canny_high": 160, "num_windows_y": 18, "num_windows_x": 34, "max_score": 2.967309236526489 3, "mean_score": 0.08812437951564789}}</tool_response>

<tool_call>{"name": "**zoom_image**", "arguments": {"image_index": 0, "x1": 520, "y1": 320, "x2": 620, "y2": 420, "scale": 3.0}}</tool_call>

<tool_response>{"status": "success", "images": "<image>", "output": {"box": [520, 320, 619, 420], "scale": 3.0}}</tool_response>

<think>The edge anomaly detection on image_index 2 (1.15s) shows a high-scoring box at [600,831,657,935] which corresponds to the logo area, similar to the previous frame but slightly shifted. More importantly, comparing the two frames, the object the person is holding has changed shape significantly between 0.85s and 1.15s. The first frame shows a flat item while the second shows a different, possibly rotated or replaced object. This rapid change in the held object across consecutive frames suggests tampering. The transition likely starts just before 0.85s and ends just after 1.15s. I should check the exact boundaries by examining the transition more closely. I'll extract a very narrow segment around 0.8s to 1.2s to get higher temporal resolution and confirm the exact start/end of the tampering.<think>

<tool_call>{"name": "**extract_video_segment_frame**", "arguments": {"start_sec": 0.8, "end_sec": 1.2}}</tool_call>

<tool_response>{"status": "success", "images": "<image><image>", "output": {"time_range_sec": [0.8, 1.2], "raw_frame_range": [16, 24], "raw_frame_count_in_range": 9, "sampled_frame_count_for_mllm": 2, "sampled_frame_indices_for_mllm": [16, 24], "sampled_frame_timestamps_sec_for_mllm": [0.8, 1.2], "sampled_fps_for_mllm": 4.444444, "sampled_image_count": 2}}</tool_response>

<think>The new extraction gives me frames at exactly 0.8s and 1.2s. Comparing these with the previous frames: at 0.8s the person is holding a flat item (similar to 0.85s), while at 1.2s the object has changed to the rotated/longer shape seen at 1.15s. This confirms the transition happens between 0.8s and 1.2s. The earlier edge anomalies were consistent with tampering in the object region during this interval. Since the change is abrupt and localized to the held object, this segment is likely tampered. The boundaries should be from just before the change starts (0.8s) to just after it completes (1.2s). Given the frame timestamps, the tampered segment is 0.80s-1.20s.<think>

<answer>**Answer: ["0.80s-1.20s"]**<answer>

Figure 8: Case study 4.

Case5 – Segmentation

Question: Please locate the tampered region and use the SAM3 segmentation tool to generate the final mask.



OmniVL-Guard Pro:

<think>I notice a large region in the image that appears to contain a painting — it's visually distinct from the background and foreground elements. To assess whether this area has been tampered with, I'll start by examining its edges and internal structure for inconsistencies. I'll zoom into this region to get a closer look at potential anomalies.</think>



<tool_call>{"name": "**zoom_image**", "arguments": {"image_index": 0, "x1": 85, "y1": 143, "x2": 918, "y2": 738, "scale": 2.0}}</tool_call>

<tool_response>{"status": "success", "images": "<image>", "output": {"box": [85, 143, 918, 738], "scale": 2.0}}</tool_response>

<think>After zooming in, I observe that the painting contains two eagles perched on branches — one near the top and another near the bottom. The top eagle's head appears unusually smooth and lacks fine feather detail, which may suggest digital manipulation. To investigate further, I'll analyze the edges within this region for irregularities that could indicate tampering.</think>

<tool_call>{"name": "**detect_edge_anomaly**", "arguments": {"image_index": 0, "block_size": 32, "stride": 16, "canny_low": 50, "canny_high": 150, "top_k": 5}}</tool_call>



<tool_response>{"status": "success", "images": "<image>", "output": {"suspicious_boxes": [{"score": 3.427011489868164, "box": [867, 469, 914, 500], "edge_density": 0.23828125, "lap_energy": 48.861328125, "neighbor_gap": 8.59429931640625}, {"score": 3.1179940700531006, "box": [703, 750, 750, 781], "edge_density": 0.3017578125, "lap_energy": 45.984375, "neighbor_gap": 5.92034912109375}, {"score": 3.0251986980438232, "box": [258, 875, 305, 906], "edge_density": 0.26953125, "lap_energy": 41.890625, "neighbor_gap": 6.68817138671875}, {"score": 2.9954140186309814, "box": [422, 828, 469, 859], "edge_density": 0.3369140625, "lap_energy": 54.029296875, "neighbor_gap": 3.7099609375}, {"score": 2.9882943630218506, "box": [679, 875, 726, 906], "edge_density": 0.2861328125, "lap_energy": 41.369140625, "neighbor_gap": 6.1661376953125}], "global_stats": {"block_size": 32, "stride": 16, "canny_low": 50, "canny_high": 150, "num_windows_y": 63, "num_windows_x": 41, "max_score": 3.427011489868164, "mean_score": 0.27704912424087524}}}</tool_response>

<think>The edge analysis reveals several suspicious areas, with the strongest signal located near the top eagle's head. This region shows subtle discontinuities in texture and edge continuity, reinforcing the suspicion of manipulation. To better understand the nature of the tampering, I'll now examine the facial structure of the top eagle more closely — if it's a real bird, we'd expect to see clear facial features; if not, it may be a synthetic insertion.</think>

<tool_call>{"name": "**detect_faces**", "arguments": {"image_index": 0, "det_size": 640, "score_threshold": 0.5, "max_faces": 5}}</tool_call>

<tool_response>{"status": "success", "output": {"bboxes": []}}</tool_response>

<think>The face detection tool returned no results, which is expected since eagles don't have human-like faces. However, the absence of detectable facial features doesn't rule out tampering — it just means the manipulation may not involve facial structure. Instead, I'll focus on the top eagle's head, where the edge anomalies were most pronounced. I'll zoom in on that specific area to inspect for signs of blending, texture mismatch, or unnatural smoothing.<think>



<tool_call>{"name": "**zoom_image**", "arguments": {"image_index": 0, "x1": 520, "y1": 320, "x2": 620, "y2": 420, "scale": 3.0}}</tool_call>

<tool_response>{"status": "success", "images": "<image>", "output": {"box": [520, 320, 619, 420], "scale": 3.0}}</tool_response>

<think>The close-up of the top eagle's head confirms my suspicion: the texture is unnaturally smooth, lacking the fine feather detail and subtle shading you'd expect in a real painting. The edges around the head also appear slightly blurred, suggesting it may have been digitally inserted or altered. Based on this, I'll define a bounding box covering the entire painting region, place a positive point on the top eagle's head — the most suspicious area — and add negative points on the background within the same box to help SAM3 distinguish the tampered region from its surroundings.<think>



<tool_call>{"name": "sam3_segmentation", "arguments": {"image_index": 0, "input_boxes": [[[85, 143, 918, 738]]], "input_points": [[[[600, 380], [100, 200], [800, 600]]]], "input_labels": [[[1, 0, 0]]]}}</tool_call>

<tool_response>{"status": "success", "images": "<mask_image>", "output": {"mask_file": "test30k_033690_sam3_mask.png"}}</tool_response>

<answer>**Answer: Segmentation complete. Mask generated as png.**<answer>

Figure 9: Case study 5.