FORENSICCPN: TOWARDS SELF-EVOLVING FORENSIC SKILLS FOR IMAGE FORGERY DETECTION AND LOCALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal large language models (MLLMs) with external forensic tools can combine semantic reasoning with specialized evidence for image forgery detection and localization. Existing methods, however, largely encode tool-use experience in model parameters or predefined workflows, rather than learning it as a directly revisable forensic skill. This limits the ability to directly update forensic experience when new attacks and tools emerge. We propose FORENSICCPN, which treats the executable forensic skill as the learning target while keeping the MLLM and expert tools frozen. Inspired by Colored Petri Nets (CPNs), FORENSICCPN represents forensic experience as an executable skill graph. The agent first proposes a hypothesis about the manipulation and acquires evidence to verify it. As evidence accumulates, the hypothesis is confirmed, revised, or resolved into a verdict, with the evolving context carried through the CPN. A forensic harness executes the skill by maintaining its evolving evidence state and enforcing graph-defined actions. The skill is self-evolving through both skill learning and test-time skill evolving: multiple agents distill labeled execution trajectories into validated revisions, while unresolved cases trigger label-free exploration whose reusable discoveries are consolidated back into the skill. Across seven conventional manipulation benchmarks, FORENSICCPN achieves the strongest aggregate detection performance while maintaining competitive region-localization performance among the compared methods. On AI inpainting and global synthesis, test-time skill evolving further improves adaptation without parameter updates.

1 INTRODUCTION

Rapid advances in image editing and generation have made image forgery detection and localization (IFDL) increasingly important (Bunk et al., 2017; Asghar et al., 2017; Guo et al., 2023). Specialized neural networks detect manipulation by exploiting noise fingerprints (Zhu et al., 2024), fusing multimodal features (Liu & Yao, 2024), and adapting visual foundation models (Peng et al., 2025). The semantic understanding and reasoning abilities of multimodal large language models (MLLMs) further provide a basis for organizing forensic evidence and explaining detection decisions (Xu et al., 2025). However, MLLMs still have limited perception of low-level cues such as compression inconsistencies, noise residuals, and frequency anomalies (Guo et al., 2026).

Recent studies have equipped MLLM agents with expert detectors and visual forensic operators as tools, enabling the agents to transform forgery traces that are difficult to observe into explicit evidence, improving forgery detection performance through planning, tool calling, and reasoning. These approaches can be grouped into two categories. (1) **Agentic Reinforcement Learning** (Zhang et al., 2025; Wang et al., 2026; Zhu et al., 2026) trains models to adaptively select and combine tools by providing reward signals related to tool invocation and evidence analysis; although this enables adaptive tool use, the learned policies are typically encoded in model parameters, making them difficult to inspect or revise without retraining. (2) **Frozen-MLLM Tool Use** (Liang et al., 2025; Yu et al., 2026) freezes MLLM parameters and guides MLLMs to use tools through external knowledge such as predefined workflows, expert profiles, and historical cases, enabling multi-source evidence analysis and decision at inference time; yet these procedures are largely predefined and static, primarily guiding execution rather than being learned and revised from execution.

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.

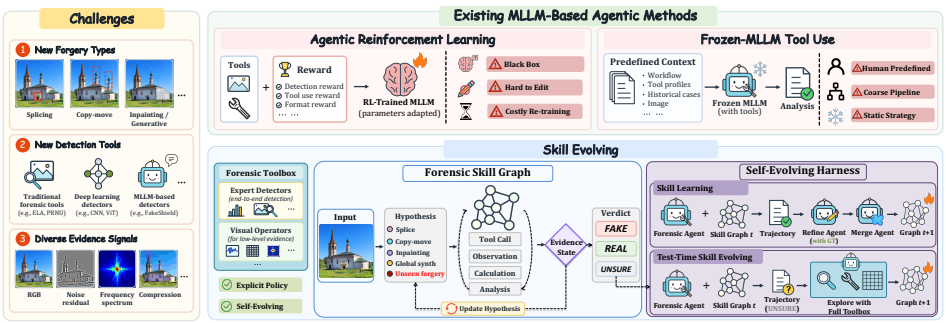


Figure 1: Motivation. Left: New forgery types, forensic tools, and evidence modalities continually change effective forensic strategies. Upper right: Existing MLLM-based agents either encode tool-use policies in model parameters or rely on predefined external guidance, making strategy updates depend on retraining or manual redesign. Lower right: FORENSICCPN externalizes forensic experience as an evidence-driven executable skill graph, which is executed and revised through self-evolving.

These limitations are particularly prominent in IFDL scenarios. As illustrated in Figure 1, the emergence of new **forgery types**, advances in **forensic tools**, and heterogeneity in **evidence signals** can change tool selection and decision rules. Therefore, as new feedback arrives, the system needs to accumulate effective forensic experience, reuse strategies in similar situations, and revise the conditions under which that experience applies. This motivates an external experience representation that distills successful trajectories and failed attempts into inspectable forensic rules.

Recent advances in agent skills provide a way to externalize reusable experience (Xu & Yan, 2026; Zhou et al., 2026b). Existing structured skill representations, such as SkillDAG (Bai et al., 2026) and GraSP (Xia et al., 2026b), organize tool dependencies and execution paths. Image forensics, however, requires more than organizing tool execution: each tool provides partial evidence rather than a decisive result, so the next action should depend on the evidence accumulated so far, not merely on static tool dependencies. This calls for a skill representation that carries the current evidence state and lets it govern transitions between actions. Inspired by Colored Petri Nets (CPNs) (Gehlot & Nigro, 2010), we propose FORENSICCPN, an agent framework that represents forensic experience as an executable skill graph. CPNs are state-transition models in which structured tokens carry execution context and transitions are guarded by explicit conditions, naturally supporting evidence-driven execution. The agent first proposes a hypothesis about manipulation or authenticity and acquires evidence to investigate it along the CPN-defined procedure. In this way, forensic experience is externalized as an evidence-governed skill whose decision logic can be directly inspected and revised.

Executing such an evidence-governed skill requires more than exposing the graph to a general agent. We design a forensic harness that maintains the evolving evidence state and constrains the agent to graph-defined actions. At each step, the agent performs the prescribed action and interprets the acquired evidence, while an independent judge verifies whether the corresponding transition condition is satisfied. The harness then advances execution according to the verification result. In this way, the skill graph acts as an executable decision procedure beyond providing contextual guidance.

The explicit skill representation makes the forensic procedure analyzable through execution trajectories, allowing observed successes and failures to be translated into revisions of the skill. Based on this, we design two trajectory-driven modes of self-evolving: skill learning and test-time skill evolving. During skill learning, multiple agents use labeled trajectories to diagnose procedural defects and propose revisions, which are verified through paired validation before updating the skill. At test time, unresolved trajectories trigger label-free exploration beyond the current skill, and reusable discoveries are consolidated back into the skill for subsequent cases.

Our contributions can be summarized as follows:

❑ **Evidence-driven forensic skill.** Inspired by CPNs, we represent forensic experience as an executable skill graph whose evolving evidence state governs tool execution and forensic decisions.

❑ **Execution harness for forensic skill.** We design a dedicated harness that maintains the evidence state, enforces graph-defined actions, and independently verifies transition conditions, turning the skill graph into an executable decision procedure.

❑ **Self-evolving for forensic skill.** We develop two modes of self-evolving based on execution trajectories: skill learning converts labeled trajectory feedback into validated skill revisions, while test-time skill evolving explores unresolved cases without labels.

❑ Experiments across seven conventional manipulation benchmarks show that FORENSICCPN achieves the strongest aggregate detection performance while maintaining competitive localization performance. Experiments on AI inpainting and global synthesis further demonstrate effective test-time adaptation.

2 RELATED WORK

Agentic skills. Agent skills externalize procedures and tool knowledge into reusable capability units. AWM (Wang et al., 2024) accumulates code skills and workflows from interaction experience. EvoSkill (Alzubi et al., 2026) and Memento-Skills (Zhou et al., 2026a) revise external skill libraries using feedback, whereas SkillRL (Xia et al., 2026a) jointly evolves skills and the model policy. Structured approaches such as SkillDAG (Bai et al., 2026) model dependencies, compositions, and conflicts; GraSP (Xia et al., 2026b) compiles skills into acyclic execution graphs with preconditions and local repair. In contrast, FORENSICCPN makes accumulated forensic evidence part of the skill state, allowing evidence to govern both execution and revision of the procedure.

Agents for image forensics. Tool-augmented forensic agents combine semantic reasoning with specialized low-level signals. ForenAgent (Zhang et al., 2025), ForgeryVCR (Wang et al., 2026), and EvoGuard (Zhu et al., 2026) use post-training to learn code-based analysis, visual-centric reasoning, and dynamic tool orchestration. AIFo (Liang et al., 2025) and AgentFoX (Yu et al., 2026) instead freeze the language model and organize evidence through multi-agent collaboration, historical cases, or expert reliability profiles. ClueAegis (Cao et al., 2026) learns tool-assisted reasoning within a predefined skill space, while ForeAgent (Wu et al., 2026) improves model parameters through label-guided reflection and trajectory regeneration. FORENSICCPN evolves the external forensic procedure itself from execution trajectories through skill learning and test-time skill evolving, while keeping the MLLM and forensic tools frozen.

3 METHODOLOGY

3.1 OVERVIEW

FORENSICCPN consists of an executable forensic skill graph, a dedicated execution harness, and two trajectory-driven modes of self-evolving, as illustrated in Figure 2. Inspired by Colored Petri Nets (CPNs) (Gehlot & Nigro, 2010), the skill externalizes forensic experience as a stateful graph whose execution context carries the current hypothesis and accumulated evidence, while evidence conditions govern transitions between actions. The forensic harness maintains this evolving state, constrains the agent to graph-defined actions, and uses an independent Judge to verify transition conditions. Because the skill is explicit and executable, its behavior can be analyzed through execution trajectories and directly revised: labeled trajectories support skill learning through paired validation, while unresolved test-time cases trigger label-free exploration whose reusable discoveries are written back into the skill.

These design choices are motivated by the challenges in Figure 1. Heterogeneous forensic signals are organized within an evolving evidence state. By decoupling tool capabilities from skill logic, the same skill representation can reorganize its evidence paths as stronger or newly available tools enter the toolbox. Skill learning and test-time skill evolving further adapt these paths when new forgery types expose gaps in the current procedure.

3.2 PRELIMINARY

Task formulation. Given an image x, the task is to determine whether it is forged and, when manipulation is present, to localize the manipulated region. We represent the task output as

$$\mathcal{F}(x) = (v, R, E), \tag{1}$$

where $v \in \{\texttt{FAKE}, \texttt{REAL}, \texttt{UNSURE}\}$ denotes the verdict, R denotes the reported manipulation region when applicable, which is defined as the entire image for global synthesis, and E denotes the accumulated forensic evidence supporting the verdict. We set UNSURE when the available evidence is insufficient to support a definite verdict.

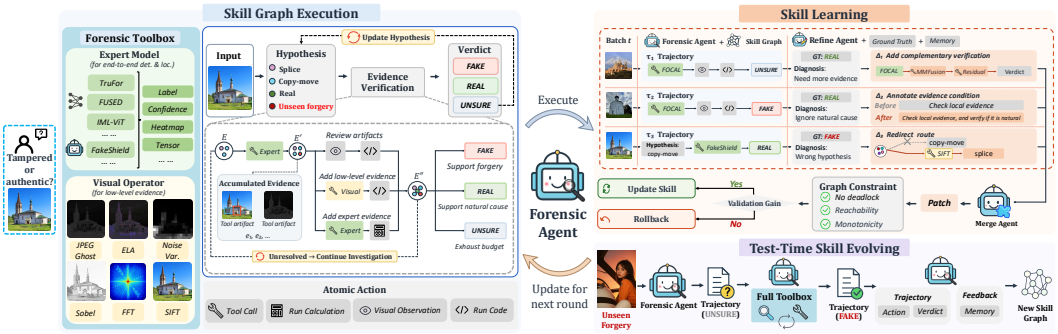


Figure 2: Overview of FORENSICCPN. Left: the forensic toolbox supplies evidence, while the skill graph and harness govern evidence-driven execution. **Upper right**: skill learning turns labeled trajectories into validated skill revisions. **Lower right**: test-time skill evolving explores beyond an UNSURE outcome and writes reusable discoveries back into the skill.

Tool selection. The agent's forensic capability is bounded by the evidence provided by its tools, making the coverage and reliability of the toolbox critical to subsequent skill execution. We first collect a broad set of deployable forensic expert models and visual operators covering complementary forensic cues. Candidate expert models are then evaluated on initialization data that is disjoint from subsequent skill learning and evaluation, and ineffective candidates are removed. The resulting base toolbox contains 11 conventional expert models, 3 MLLM-based forensic models, and 13 primitive visual operators. Its trained models and primitive operators remain fixed during subsequent skill execution and evolution. Details of toolbox composition are provided in Appendix A.4.

3.3 FORENSIC SKILL

Motivation. Unlike general tool-use procedures, forensic execution must reason over partial evidence. We therefore organize the forensic investigation around a falsifiable hypothesis: the agent first proposes a hypothesis about manipulation or authenticity, then acquires evidence to test it, and either strengthens the hypothesis, revises it when contradicted, or reaches a verdict when the evidence becomes sufficient. We refer to the hypothesis-specific investigation as a forensic branch.

This forensic paradigm requires a skill representation that captures not only the current execution step, but also the current hypothesis and the evidence. The same forensic branch may call for different next actions under different evidence states, and when new evidence contradicts the current hypothesis, execution must return to hypothesis formation without discarding what has already been observed. A representation that carries evolving context and lets that context govern state transitions is necessary. Colored Petri Nets (CPNs) (Gehlot & Nigro, 2010) provide this mechanism naturally: places represent execution states, transitions represent state changes, arcs connect them and define allowed flow, and colors specify the typed context carried by tokens. As tokens move through the graph, their content is updated, and the current context determines which transitions can fire next.

Formulation. In our CPN formulation, we represent a skill graph as

$$G = (P, T, A, \mathcal{C}), \qquad \xi_t = (x, h_t, R_t, E_t) \in \mathcal{C}, \tag{2}$$

where P is the set of forensic states, T is the set of transitions associated with forensic actions, A denotes directed arcs between places and transitions together with their evidence conditions, and $\mathcal{C}$ is the space of typed execution contexts carried by tokens. At step t, the execution token ξ_t carries the input image x, the current hypothesis h_t, the candidate region R_t, and the accumulated evidence E_t. The current execution state is determined jointly by the token's position in the graph and its structured context. The hypothesis h_t determines the forensic question to be tested and the evidence relevant to the current branch. Executing a transition acquires new evidence and updates the execution context ξ_t. The updated evidence state then determines whether execution continues within the current branch, returns to hypothesis formation when the hypothesis is no longer supported, or terminates with a verdict. In this way, the skill specifies an evidence-conditioned forensic procedure instead of a fixed sequence of tool calls. At a terminal state, the terminal place determines the verdict v, while the final R_t and E_t provide the reported region and supporting evidence in Eq. (1).

Skill initialization. We initialize the skill G_0 with three forensic branches for copy-move, splicing, and authenticity. TruFor provides shared evidence for scouting before hypothesis formation because of its rich outputs and strong overall performance on the initialization data. Each branch is then seeded with one additional tool selected for its observed branch-specific effectiveness. This

lightweight initialization provides a viable starting procedure that is subsequently expanded and revised through skill learning. Details of the initialization data and tool screening are provided in Appendix A.4.

3.4 Harness for Forensic Skill

Motivation. A general agent harness can manage interaction context and expose tools (Liang et al., 2026), but these functions alone do not ensure that an evidence-conditioned skill is followed. Our forensic harness extends this role by maintaining the current CPN state, organizing the evidence exposed at each step, and restricting the available operations according to the skill. Most importantly, graph-defined actions and transition conditions are enforced throughout execution rather than left to the agent's discretion.

Execution. At step t, the harness maintains the current place p_t and execution token ξ_t, and constructs the context available to the Forensic Agent. The agent performs the prescribed action and interprets the resulting evidence. The Judge evaluates numerical transition conditions directly from the recorded measurements and uses an MLLM to verify semantic conditions against the agent's assessment and cited evidence. The harness then advances along the verified arc or the graph-defined fallback and updates the execution state. This turns the skill graph from contextual guidance into an executable procedure whose evidence state and transition logic reliably govern agent behavior. Context organization, evidence-access operations, and runtime budgets are detailed in Appendix A.4.

3.5 Skill Learning

With the harness above, the forensic skill is executed as an explicit evidence-governed procedure, making its behavior directly observable through execution trajectories. Skill learning uses these trajectories together with ground-truth feedback to diagnose procedural defects and translate them into reusable revisions of the skill.

Specifically, skill learning proceeds through three agents with complementary roles.

Forensic Agent. Starting from the current skill G_k, the Forensic Agent executes each training image x_i without access to its label and records the resulting trajectory $\tau_i^k = \mathrm{Exec}(G_k, x_i)$. Since the execution policy is fixed by the current skill graph, the trajectory provides a faithful record of how the skill behaves on that sample and serves as the basis for diagnosing the skill itself. For a training batch B_k, these sample-level executions are independent and can be followed by parallel refinement.

Refine Agent. The Refine Agent uses the labeled trajectory together with revision memory M_k to determine whether the execution exposes a procedural defect and localize the responsible part of the skill. The memory records only previously rejected revision structures and their validation history, helping avoid repeated unsuccessful directions. The Refine agent then proposes a local revision

$$\Delta_i^k = \mathrm{Refine}(\tau_i^k, y_i, G_k, M_k),$$

which may modify tool use, routing, execution guidance, or evidence conditions. When a recurring computation over acquired evidence proves useful, it may also be encapsulated as a reusable derived tool. If no generalizable defect is identified, the Refine agent returns $\Delta_i^k = \varnothing$.

Merge Agent. Local revisions proposed from individual trajectories may be sample-specific or mutually incompatible. The Merge Agent reviews the batch-level proposals together with revision memory M_k, combines compatible revisions that address the same procedural defect, and forms a skill-level candidate

$$\Delta^k = \mathrm{Merge}(\{\Delta_i^k\}_{i\in B_k}, M_k), \qquad G_k' = \mathrm{Apply}(G_k, \Delta^k).$$

The resulting candidate preserves a coherent modification of the shared skill while excluding redundant or conflicting local proposals.

Paired Validation. A skill-level candidate is first checked for structural validity, including valid dependencies, reachable routes, and consistent evidence conditions. Structurally valid candidates are then evaluated against the current skill on the same held-out validation cohort V_k. Let $\ell_i(G) \in \{\mathrm{FAKE}, \mathrm{REAL}\}$ denote the binary leaning of an uncertain verdict. Each sample receives the utility

$$q_i(G) = \mathbf{1}[v_i(G) = y_i] + (1-\lambda)\mathbf{1}[v_i(G) = \texttt{UNSURE} \wedge \ell_i(G) = y_i], \qquad \lambda = \frac{1}{2}, \tag{3}$$

where correct definite verdicts receive full credit, correct uncertain leanings receive discounted credit, and incorrect decisions receive zero. We then measure the paired utility gain

$$\Delta Q_k = \frac{1}{|V_k|} \sum_{i \in V_k} \left[q_i(G'_k) - q_i(G_k) \right], \qquad G_{k+1} = \begin{cases} G'_k, & \Delta Q_k > 0, \\ G_k, & \text{otherwise.} \end{cases} \tag{4}$$

Only rejected revision structures and their validation history update the revision memory M_{k+1}, which informs subsequent Refine and Merge agents without affecting execution-time routing. Through this cycle, execution experience is progressively distilled into validated and reusable revisions of the skill graph.

3.6 Test-Time Skill Evolving

Motivation. A learned skill captures forensic strategies distilled from trajectories, but unseen attacks or unfamiliar evidence combinations at test time may fall outside its current execution paths. This limitation concerns the coverage of the learned skill rather than the capacity of the forensic toolbox, which may still contain tools capable of resolving such cases. Under evidence-governed execution, this gap becomes explicit when the prescribed paths cannot satisfy the conditions for a definite verdict and terminate at `UNSURE`. We use these cases to trigger test-time skill evolving, allowing the agent to explore beyond the current skill and accumulate reusable experience for subsequent cases.

Exploration. For each triggered case, the agent retains the final execution token $\xi_i^\star = (x_i, h_i, R_i, E_i)$ together with the unresolved condition that led to `UNSURE`. It then leaves the current skill and performs targeted exploration with the full toolbox. Guided by the unresolved condition and the evidence already acquired, the agent selects additional tools and analyses that can address the missing forensic question, rather than exhaustively invoking the toolbox. Exploration continues until the additional evidence supports a definite verdict or no useful progress can be made. Successfully resolved cases are summarized as exploration experiences ϵ_i, which combine the original UNSURE trajectory, the newly acquired evidence, and the resolved verdict. Each experience captures why the current skill stalled and what additional evidence enabled further progress, but does not directly modify the shared skill.

Skill adaption. Cases within a test-time batch are processed independently under the same frozen skill G_k, and skill revision occurs only at the batch boundary. Let $\mathcal{E}_k$ denote the successfully resolved exploration experiences in the batch, $\mathcal{D}_k$ the cases that reached definite verdicts under G_k, and M_k the cross-batch memory. A batch-level consolidation agent produces

$$(\Delta_k, M_{k+1}) = \text{Consolidate}(\mathcal{E}_k, \mathcal{D}_k, M_k),$$

where Δ_k is at most one reusable skill revision and M_{k+1} carries useful context to subsequent batches. Consolidation does not use ground-truth labels or paired validation; a non-empty revision is written back as $G_{k+1} = \text{Apply}(G_k, \Delta_k)$ only after satisfying the structural and execution constraints of the skill. Otherwise, the skill remains unchanged.

4 Experiments

4.1 Experimental Setup

Datasets. We use separate datasets for toolbox screening, skill learning, and evaluation. Tamp-COCO (Kwon et al., 2021) and SID-Set (Huang et al., 2025) are used for toolbox screening and skill initialization, while CASIA V2 (Dong et al., 2013) and IMD2020 (Novozámskỳ et al., 2020) provide labeled data for skill learning. We evaluate conventional IFDL performance on CASIA V1 (Dong et al., 2013), Columbia (Hsu & Chang, 2006), COVERAGE (Wen et al., 2016), DSO-1 (De Carvalho et al., 2013), Korus (Korus & Huang, 2016), In-the-Wild (Huh et al., 2018), and NIST16 (Guan et al., 2019). Test-time skill evolving is evaluated on COCOGlide (Guillaro et al., 2023) and AIGI-Bench (Li et al., 2026), whose generative attacks are unseen during skill learning. Dataset composition and sample counts are provided in Appendix A.1.

Baselines. We compare against specialist IFDL models (Bai et al., 2025a; Chen et al., 2021; Liu et al., 2022; Peng et al., 2025; Guillaro et al., 2023; Triaridis & Mezaris, 2024; Kwon et al., 2021; Ma et al., 2023; Wu et al., 2025), and general and forensic MLLMs (Bai et al., 2025b; OpenAI, 2026; Huang et al., 2025; Xu et al., 2025; Wen et al., 2026; Wang et al., 2026). We additionally construct Majority Voting, logistic regression, and MLP baselines to test whether static fusion of toolbox predictions can explain the gains of ForensicCPN, together with a dataset-level oracle single-tool reference. Detailed baseline interfaces and configurations are provided in Appendix A.3.

Table 1: Image-level forgery detection (%). W. Avg. weights datasets by image count. Bold/underlining mark the best/second-best distinct values, excluding the oracle; displayed ties share a rank.

Method	CASIA v1		Columbia		Coverage		DSO-1		In-the-Wild		Korus		NIST16		W. Avg.	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Expert forensic models																
PIM	69.5	75.0	57.0	69.4	49.5	65.5	52.5	66.9	95.5	97.7	51.6	65.7	41.5	55.9	57.5	67.9
MVSS-Net	76.0	73.2	67.8	75.5	55.0	66.9	47.0	64.0	96.5	98.2	53.9	63.2	41.7	55.1	61.1	67.1
PSCC-Net	74.4	71.3	48.8	65.6	61.5	62.1	61.0	54.7	32.3	48.9	50.2	66.8	46.6	57.2	58.3	63.8
ForensicsSAM	70.5	70.5	98.6	98.6	58.5	68.0	80.5	80.8	91.5	95.6	66.1	53.0	78.0	67.6	75.5	71.6
TruFor	82.7	83.0	94.8	95.0	68.5	63.6	86.5	87.9	75.6	86.1	67.5	55.2	67.4	55.0	76.6	71.9
MMFusion-IML	85.4	85.8	90.1	90.9	76.5	75.7	87.5	88.2	77.6	87.4	72.7	66.1	69.5	58.2	78.9	75.3
MLLM-based methods																
Qwen3.5-397B-A17B	80.5	77.7	96.0	96.0	51.0	18.3	50.5	2.0	74.0	85.1	54.0	17.9	49.5	26.9	66.5	51.8
GPT-5.6-Luna	47.5	29.8	84.0	83.0	43.0	2.0	50.0	3.9	45.0	62.1	50.0	13.1	49.0	33.3	50.9	32.6
SIDA	59.2	46.2	87.3	87.8	50.0	16.7	51.5	7.6	64.7	78.6	53.0	20.7	64.6	33.3	62.0	41.4
FakeShield	87.9	88.8	91.5	91.7	55.0	63.1	67.0	73.8	66.2	79.6	55.9	44.6	70.4	61.4	76.3	73.9
FakeVLM	45.2	9.1	22.0	5.4	51.0	55.1	61.0	42.7	37.8	54.9	51.4	55.4	61.0	51.8	49.6	32.2
ForgeryVCR	89.5	90.1	90.9	91.6	71.5	71.4	82.0	79.1	92.5	96.1	61.8	49.7	73.3	66.4	80.8	77.8
Tool ensembles and oracle reference																
Majority Voting	84.7	83.9	96.1	96.2	73.0	73.5	92.5	92.1	75.6	86.1	69.3	58.5	77.5	68.4	81.3	77.5
LogReg	66.5	73.4	58.1	70.2	70.5	64.7	**95.5**	**95.3**	76.6	86.8	64.8	63.9	65.1	61.7	67.1	69.7
MLP	67.8	71.6	84.8	86.7	74.0	71.1	86.5	84.4	76.6	86.8	68.6	58.7	74.0	64.8	72.7	70.6
Oracle single tool	87.9	88.8	98.6	98.6	76.5	75.7	87.5	88.2	96.5	98.2	72.7	66.1	78.0	67.6	84.0	80.5
Executable forensic skills																
FORENSICCPN (init.)	77.0	73.6	77.0	75.1	63.0	49.3	73.0	63.0	66.0	79.5	59.0	36.9	59.0	42.9	68.3	59.2
FORENSICCPN (evolved)	**92.0**	**92.9**	**99.5**	**99.5**	**81.0**	**79.6**	93.0	93.3	**97.7**	**98.8**	**77.5**	**74.9**	**83.7**	**82.8**	**88.4**	**88.2**

Metrics. We report image-level accuracy and F1 for detection, and box IoU (B-IoU) before segmentation refinement for localization. For evaluation, an `UNSURE` verdict is resolved using its binary leaning, without the utility discount used for skill admission. Forged images predicted as `REAL` or without a valid localization receive zero B-IoU. Weighted averages use all images for detection and only forged images for localization. Detailed definitions are provided in Appendix A.2.

Implementation. Unless otherwise stated, all MLLM-based agent roles use Qwen3.6-35B-A3B with a decoding temperature of 0. Detailed runtime and skill evolving settings are provided in Appendix A.4.

4.2 OVERALL PERFORMANCE

Detection. Table 1 reports image-level detection across seven conventional benchmarks, where FORENSICCPN achieves the strongest aggregate performance over specialist, MLLM-based, and multi-tool fusion baselines. Its advantage over static fusion shows that the gain cannot be explained by prediction aggregation alone, but relies on evidence-guided execution. The further improvement from the initialized to the evolved skill, with the MLLM and forensic tools frozen, supports learning and revising the external forensic skill rather than parameter updates.

Table 2: Region-level localization by B-IoU (%) before segmentation refinement. W. Avg. uses manipulated-image counts. Columns follow the ranking convention of Table 1.

Method	CASIA v1	Columbia	Coverage	DSO-1	In-the-Wild	Korus	NIST16	W. Avg.
TruFor	57.5	51.9	38.0	75.7	35.4	23.7	22.9	43.3
CAT-Net	60.4	77.1	18.7	24.1	8.0	0.1	14.6	36.6
IML-ViT	52.5	48.9	20.6	14.1	20.7	16.0	24.3	35.9
MMFusion-IML	65.7	59.3	49.5	64.5	42.8	26.5	29.3	49.7
FOCAL	**75.8**	**96.4**	**58.9**	40.7	48.2	25.0	38.8	58.7
ForensicsSAM	52.8	87.3	36.2	71.6	58.0	22.1	49.9	52.4
FakeShield	54.5	78.4	23.1	43.6	46.3	10.3	22.6	41.7
ForgeryVCR	69.7	91.5	57.4	47.4	60.2	15.2	39.5	56.4
FORENSICCPN	74.1	75.3	52.1	**81.9**	**64.4**	**35.7**	**50.3**	**63.1**

Localization. Table 2 evaluates region-level localization. FORENSICCPN achieves the highest weighted B-IoU, showing that evidence-driven execution also provides strong spatial grounding. Columbia is a notable exception: FOCAL provides very accurate localization, whereas MMFusion is strong for classification but much weaker for localization on this dataset. Since both are used in the

learned splice branch, MMFusion can help the final classification while making the reported region less precise than using FOCAL alone. This suggests that the evidence most useful for detection is not always the evidence most useful for localization.

4.3 Skill Structure

Evidence-driven execution. Table 3 compares skill formats and execution mechanisms while fixing the backbone, toolbox, and forensic knowledge. Compared with a Markdown skill executed by general harness (Tencent Cloud, 2026; OpenAI, 2025), CPN achieves higher accuracy with fewer rounds and tokens, while reducing the UNSURE rate to 1.8%, showing the benefit of explicit evidence states and controlled transitions. Under the same harness, DAG is cheaper but less accurate and produces more UNSURE cases as it doesn't perform hypothesis revision. In contrast, CPN can revise the hypothesis when evidence conflicts while retaining prior observations, supporting stateful forensic execution. Detailed settings are provided in Appendix A.3.

Table 3: Skill structure and cost. Fixed backbone, tools, and skill content. Accuracy is in %.

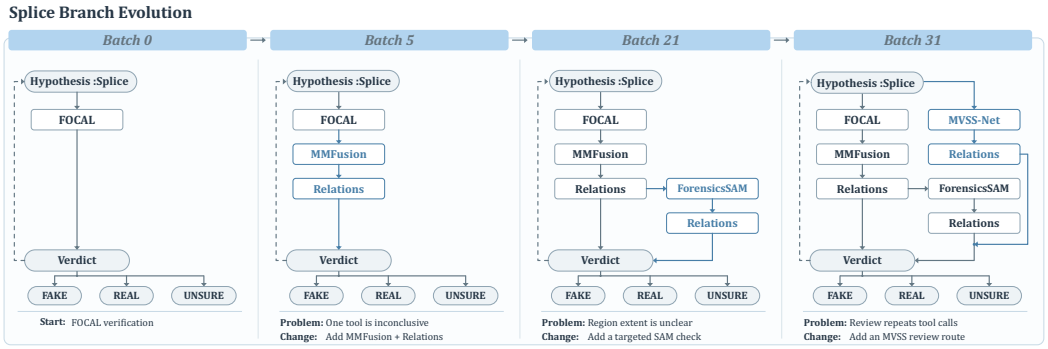
Skill format	Markdown		DAG	CPN
Harness	Codex	Codebuddy	Forensic	Forensic
CASIA v1	**92.0**	85.4	78.0	**92.0**
Columbia	**99.5**	86.2	79.6	**99.5**
COVERAGE	70.4	66.9	66.0	**81.0**
DSO-1	93.0	89.8	**97.0**	93.0
In-the-Wild	97.7	**100.0**	93.4	97.7
Korus	77.5	71.5	**78.5**	77.5
NIST16	76.4	**91.0**	80.6	83.7
Weighted Acc. ↑	85.6	85.9	80.0	**88.4**
Avg. rounds ↓	18.9	15.6	**6.8**	8.2
Avg. tokens ↓	374k	360k	**151k**	176k
UNSURE (%) ↓	35.4	37.7	16.7	**1.8**

Training comparison. Table 4 compares external skill learning with parameter-based adaptation using the same base MLLM and training data. The trained baselines predict the label and localization from the input image without access to the forensic toolbox. Under these configurations, skills achieve the highest weighted accuracy, F1, and B-IoU, supporting external skill revision as an effective alternative to parameter updates. Full results are provided in Appendix B.1.

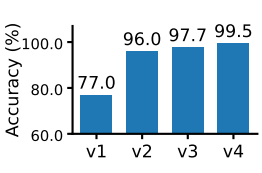
Table 4: Comparison of skills and training.

Configuration	Acc.	F1	B-IoU
Pretrained MLLM	64.8	51.5	31.1
w/ SFT	71.8	62.5	43.4
w/ SFT+GRPO	76.4	67.1	43.7
w/ CPN skills	**88.4**	**88.2**	**63.1**

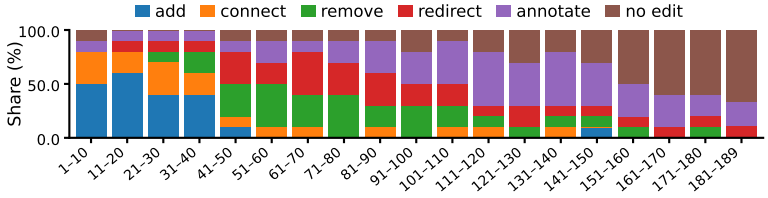
4.4 Skill Learning Analysis



(a) Splice branch revisions.



(b) Checkpoint accuracy.



(c) Refine Agent proposals across branches.

Figure 3: Skill learning. Branch revisions, checkpoint accuracy, and local proposal types. Panel (b) reports checkpoint accuracy on Columbia; panel (c) counts of proposals in skill learning.

Figure 3 traces how labeled feedback evolves the splice branch. MMFusion and relational analysis address inconclusive single-tool evidence and accompany the largest checkpoint gain. Conditional ForensicsSAM refinement then targets unclear region extent. An MVSS-Net review route reuses prior observations. Later gains accompany these more selective paths. Learning improves how evidence is acquired and used, rather than simply accumulating tools. Across branches, early `add/connect` gives way to `redirect/remove`, then `annotate` or no proposal. This progression suggests structural convergence: learning shifts from building paths to reorganizing them and refining interpretation and conditions. Refine proposes local repairs, Merge consolidates them across samples, and positive paired utility gain admits a candidate. Thus, proposal counts and checkpoint gains describe different stages of learning. A case is shown in Appendix C.2.

4.5 Test-Time Skill Evolving Analysis

Figure 4 evaluates whether test-time exploration produces reusable skill improvements. After K adaptation batches, we freeze CPN@K and reevaluate the full target set with both skill updates and free exploration disabled. The final checkpoints improve on both generative benchmarks, showing that exploration experience is consolidated into reusable CPN revisions rather than benefiting only the triggering cases.

Adaptation also expands skill coverage. The initial skill contains only copy-move, splicing, and authenticity branches, while test-time evolving introduces dedicated inpainting and global-synthesis paths for previously unseen attacks (Appendix C). Intermediate checkpoints fluctuate, indicating that individual label-free revisions are not uniformly beneficial, although the final evolved skills outperform the initial skill on both benchmarks.

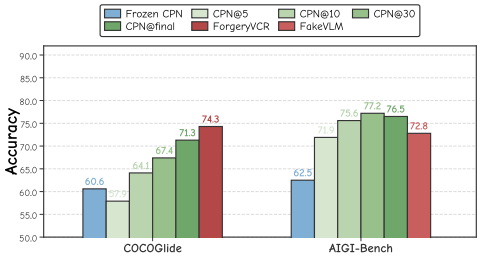


Figure 4: Test-time evolved skill.

4.6 Case Study

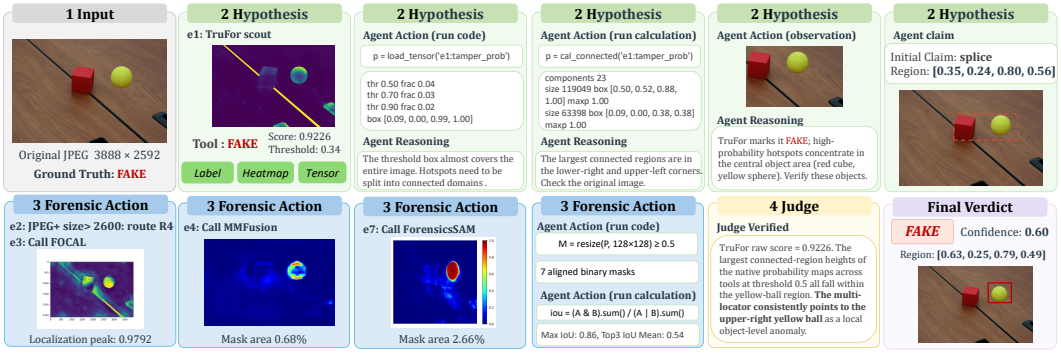


Figure 5: A NIST16 execution. Ground truth is shown only for illustration, not supplied to the agent.

Figure 5 shows how evidence changes the reported region. TruFor flags the image, but scattered responses yield an almost image-wide box. Component analysis and visual inspection motivate a splicing hypothesis covering the cube and sphere. Along the prescribed branch, FOCAL, MMFusion, and ForensicsSAM provide further evidence. Code aligns their masks, and numerical comparison concentrates support on the sphere. Condition checking then permits a `FAKE` verdict with a narrower region. The trace illustrates the division of labor: the agent interprets artifacts, while the harness carries the evolving evidence state and enforces progression.

5 Conclusion

ForensicCPN learns explicit, evidence-gated forensic skills while keeping the MLLM and expert tools frozen. Offline trajectory feedback revises complete procedures under paired selection; label-free test-time exploration turns unresolved cases into reusable updates.

AI USE STATEMENT

Generative AI assistance in preparing this manuscript included language editing and LaTeX formatting. The experimental results and case-study materials reported here were supplied by the authors. The authors are responsible for verifying the scientific claims and the final manuscript.

ETHICS STATEMENT

Image-forgery analysis can support the assessment of visual evidence, but an automated prediction should not be treated as proof of intent or provenance. False detections and inaccurate localization can affect judgments about authentic content. The framework records the evidence supporting each decision and retains an uncertain outcome when its execution conditions do not establish a definite verdict.

REPRODUCIBILITY STATEMENT

Section 3 specifies the skill representation and update criteria. Appendices A.1 and A.2 describe the data composition and evaluation protocol; Appendices A.3 and A.4 specify the comparison settings, tool interfaces, and runtime configuration. Appendix D provides the agent prompts, and Appendix C illustrates skill revisions.

REFERENCES

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Khurshid Asghar, Zulfiqar Habib, and Muhammad Hussain. Copy-move and splicing image forgery detection and localization techniques: a review. *Australian Journal of Forensic Sciences*, 49(3): 281–307, 2017.

Ningning Bai, Xiaofeng Wang, Ruidong Han, Jianpeng Hou, Yihang Wang, and Shanmin Pang. Pim-net: progressive inconsistency mining network for image manipulation localization. *Pattern Recognition*, 159:111136, 2025a.

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

Tong Bai, Zhenglin Wan, Pengfei Zhou, Xingrui Yu, Yang You, and Ivor W Tsang. Skilldag: Self-evolving typed skill graphs for llm skill selection at scale. *arXiv preprint arXiv:2606.03056*, 2026.

Jason Bunk, Jawadul H Bappy, Tajuddin Manhar Mohammed, Lakshmanan Nataraj, Arjuna Flenner, BS Manjunath, Shivkumar Chandrasekaran, Amit K Roy-Chowdhury, and Lawrence Peterson. Detection and localization of image forgeries using resampling features and deep learning. In *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pp. 1881–1889. IEEE, 2017.

Huangsen Cao, Hongkang Chu, Yuxi Li, Ying Zhang, Chen Li, Jing Lyu, Yongwei Wang, Yu Zhao, and Fei Wu. Clueaegis: Heuristic-to-reasoning cognitive-skill learning for unified evidence-based synthetic image detection. *arXiv preprint arXiv:2605.25009*, 2026.

Xinru Chen, Chengbo Dong, Jiaqi Ji, Juan Cao, and Xirong Li. Image manipulation detection by multi-view multi-scale supervision. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14165–14173. IEEE, 2021.

Tiago José De Carvalho, Christian Riess, Elli Angelopoulou, Helio Pedrini, and Anderson de Rezende Rocha. Exposing digital image forgeries by illumination color classification. *IEEE transactions on information forensics and security*, 8(7):1182–1194, 2013.

Jing Dong, Wei Wang, and Tieniu Tan. Casia image tampering detection evaluation database. In *2013 IEEE China summit and international conference on signal and information processing*, pp. 422–426. IEEE, 2013.

Vijay Gehlot and Carmen Nigro. An introduction to systems modeling and simulation with colored petri nets. In *Proceedings of the 2010 winter simulation conference*, pp. 104–118. IEEE, 2010.

Haiying Guan, Mark Kozak, Eric Robertson, Yooyoung Lee, Amy N Yates, Andrew Delgado, Daniel Zhou, Timothee Kheyrkhah, Jeff Smith, and Jonathan Fiscus. Mfc datasets: Large-scale benchmark datasets for media forensic challenge evaluation. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pp. 63–72. IEEE, 2019.

Fabrizio Guillaro, Davide Cozzolino, Avneesh Sud, Nicholas Dufour, and Luisa Verdoliva. Trufor: Leveraging all-round clues for trustworthy image forgery detection and localization. In *2023 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 20606–20615. IEEE, 2023.

Shaofeng Guo, Jiequan Cui, and Richang Hong. Rethinking vlms for image forgery detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5828–5837, 2026.

Xiao Guo, Xiaohong Liu, Zhiyuan Ren, Steven Grosz, Iacopo Masi, and Xiaoming Liu. Hierarchical fine-grained image forgery detection and localization. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3155–3165. IEEE, 2023.

Yu-Feng Hsu and Shih-Fu Chang. Detecting image splicing using geometry invariants and camera characteristics consistency. In *2006 IEEE international conference on multimedia and expo*, pp. 549–552. IEEE, 2006.

Zhenglin Huang, Jinwei Hu, Xiangtai Li, Yiwei He, Xingyu Zhao, Bei Peng, Baoyuan Wu, Xiaowei Huang, and Guangliang Cheng. Sida: Social media image deepfake detection, localization and explanation with large multimodal model. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28831–28841. IEEE, 2025.

Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A Efros. Fighting fake news: Image splice detection via learned self-consistency. In *European Conference on Computer Vision*, pp. 106–124. Springer, 2018.

Paweł Korus and Jiwu Huang. Multi-scale analysis strategies in prnu-based tampering localization. *IEEE Transactions on Information Forensics and Security*, 12(4):809–824, 2016.

Myung-Joon Kwon, In-Jae Yu, Seung-Hun Nam, and Heung-Kyu Lee. Cat-net: Compression artifact tracing network for detection and localization of image splicing. In *2021 IEEE winter conference on applications of computer vision (WACV)*, pp. 375–384. IEEE, 2021.

Ziqiang Li, Jiazhen Yan, Ziwen He, Kai Zeng, Weiwei Jiang, Lizhi Xiong, and Zhangjie Fu. Is artificial intelligence generated image detection a solved problem? *Advances in neural information processing systems*, 38, 2026.

Jiaqing Liang, Jinyi Han, Weijia Li, Xinyi Wang, Zhoujia Zhang, Zishang Jiang, Ying Liao, Tingyun Li, Ying Huang, Hao Shen, et al. Genericagent: A token-efficient self-evolving llm agent via contextual information density maximization (v1. 0). *arXiv preprint arXiv:2604.17091*, 2026.

Mengfei Liang, Yiting Qu, Yukun Jiang, Michael Backes, and Yang Zhang. From evidence to verdict: An agent-based forensic framework for ai-generated image detection. *arXiv preprint arXiv:2511.00181*, 2025.

Chengming Liu and Wenshuai Yao. Mmfa-net: Multi-modal feature learning and multi-head fuse attention for image manipulation detection and localization. In *2024 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*, pp. 1298–1302. IEEE, 2024.

Xiaohong Liu, Yaojie Liu, Jun Chen, and Xiaoming Liu. Pscc-net: Progressive spatio-channel correlation network for image manipulation detection and localization. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(11):7505–7517, 2022.

Xiaochen Ma, Bo Du, Zhuohang Jiang, Xia Du, Ahmed Y Al Hammadi, and Jizhe Zhou. Iml-vit: Benchmarking image manipulation localization by vision transformer. *arXiv preprint arXiv:2307.14863*, 2023.

Adam Novozámskỳ, Babak Mahdian, and Stanislav Saic. Imd2020: A large-scale annotated dataset tailored for detecting manipulated images. In *2020 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pp. 71–80. IEEE, 2020.

OpenAI. Codex: AI coding partner from OpenAI. Official product page, 2025. URL https://openai.com/codex/. Accessed: 2026-09-26.

OpenAI. GPT-5.6 system card. Technical report, OpenAI, July 2026. URL https://deploymentsafety.openai.com/gpt-5-6/troubleshootingbench.

Rongxuan Peng, Shunquan Tan, Chenqi Kong, Anwei Luo, Alex C Kot, and Jiwu Huang. Forensic-ssam: Toward robust and unified image forgery detection and localization resisting to adversarial attack. *arXiv preprint arXiv:2508.07402*, 2025.

Tencent Cloud. Tencent cloud codebuddy. Official product documentation, 2026. URL https://www.tencentcloud.com/document/product/1256. Accessed: 2026-07-15.

Konstantinos Triaridis and Vasileios Mezaris. Exploring multi-modal fusion for image manipulation detection and localization. In *International conference on multimedia modeling*, pp. 198–211. Springer, 2024.

Youqi Wang, Shen Chen, Haowei Wang, Rongxuan Peng, Taiping Yao, Shunquan Tan, Changsheng Chen, Bin Li, and Shouhong Ding. Forgeryvcr: Visual-centric reasoning via efficient forensic tools in mllms for image forgery detection and localization. *arXiv preprint arXiv:2602.14098*, 2026.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Bihan Wen, Ye Zhu, Ramanathan Subramanian, Tian-Tsong Ng, Xuanjing Shen, and Stefan Winkler. Coverage—a novel database for copy-move forgery detection. In *2016 IEEE international conference on image processing (ICIP)*, pp. 161–165. IEEE, 2016.

Siwei Wen, Peilin Feng, Hengrui Kang, Zichen Wen, Yize Chen, Jiang Wu, Conghui He, Weijia Li, et al. Spot the fake: Large multimodal model-based synthetic image detection with artifact explanation. *Advances in Neural Information Processing Systems*, 38:58972–59005, 2026.

Haiwei Wu, Yiming Chen, Jiantao Zhou, and Yuanman Li. Rethinking image forgery detection via soft contrastive learning and unsupervised clustering. *IEEE Transactions on Dependable and Secure Computing*, 22(6):6296–6308, 2025.

Yangjun Wu, Keyu Yan, Yu Liu, Jingren Zhou, Fei Huang, Rong Zhang, Zhou Zhao, and Fei Wu. Perception, verdict, and evolution: Hindsight-driven self-refining forensics agent for ai-generated image detection. *arXiv preprint arXiv:2606.26552*, 2026.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026a.

Tianle Xia, Lingxiang Hu, Yiding Sun, Ming Xu, Lan Xu, Siying Wang, Wei Xu, and Jie Jiang. Grasp: Graph-structured skill compositions for llm agents. *arXiv preprint arXiv:2604.17870*, 2026b.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.

Zhipei Xu, Xuanyu Zhang, Runyi Li, Zecheng Tang, Qing Huang, and Jian Zhang. Fakeshield: Explainable image forgery detection and localization via multi-modal large language models. In *International Conference on Learning Representations*, volume 2025, pp. 31186–31216, 2025.

Yangxin Yu, Yue Zhou, Bin Li, Kaiqing Lin, Haodong Li, Jiangqun Ni, and Bo Cao. Agentfox: Llm agent-guided fusion with explainability for ai-generated image detection. *arXiv preprint arXiv:2603.23115*, 2026.

Fanrui Zhang, Qiang Zhang, Sizhuo Zhou, Jianwen Sun, Chuanhao Li, Jiaxin Ai, Yukang Feng, Yujie Zhang, Wenjie Li, Zizhen Li, et al. Code-in-the-loop forensics: Agentic tool use for image forgery detection. *arXiv preprint arXiv:2512.16300*, 2025.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, et al. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026a.

Yingli Zhou, Wang Shu, Yaodong Su, Wenchuan Du, Yixiang Fang, and Xuemin Lin. A comprehensive survey on agent skills: Taxonomy, techniques, and applications. *arXiv preprint arXiv:2605.07358*, 2026b.

Chenyang Zhu, Maorong Wang, Jun Liu, Ching-Chun Chang, and Isao Echizen. Evoguard: An extensible agentic rl-based framework for practical and evolving ai-generated image detection. *arXiv preprint arXiv:2603.17343*, 2026.

Jiaying Zhu, Dong Li, Xueyang Fu, Gang Yang, Jie Huang, Aiping Liu, and Zheng-Jun Zha. Learning discriminative noise guidance for image forgery detection and localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 7739–7747, 2024.

A Experimental Settings

A.1 Datasets

Table A.1 separates the data used for skill initialization, skill learning, and evaluation. The initialization pools contain 400,000 images. Skill refinement uses 3,200 labeled images from CASIA v2 and IMD2020, while paired-validation cohorts are sampled from images in these datasets excluded from the training set. The evaluation suite contains 6,588 images across seven traditional-manipulation benchmarks and two generative-manipulation benchmarks. SID-Set's generative data are used only for toolbox screening and initialization; generative attacks are excluded from skill learning, and G_0 contains no inpainting or global-synthesis branch.

Table A.1: Dataset composition.

Dataset	Manipulation type	Fake	Real
Skill initialization			
TampCOCO (Kwon et al., 2021)	Traditional manipulation	100,000	0
SID-Set (Huang et al., 2025)	AI inpainting and full-image synthesis	200,000	100,000
Offline skill learning: refinement			
CASIA v2 (Dong et al., 2013)	Traditional manipulation	1,000	1,000
IMD2020 (Novozámskỳ et al., 2020)	Traditional manipulation	1,000	200
Traditional manipulation evaluation			
CASIA v1 (Dong et al., 2013)	Splicing and copy-move	920	800
Columbia (Hsu & Chang, 2006)	Splicing	180	183
COVERAGE (Wen et al., 2016)	Copy-move	100	100
DSO-1 (De Carvalho et al., 2013)	Splicing	100	100
In-the-Wild (Huh et al., 2018)	Traditional manipulation	201	0
Korus (Korus & Huang, 2016)	Splicing	220	220
NIST16 (Guan et al., 2019)	Splicing, copy-move, and removal	564	876
Test-time adaptation evaluation			
COCOGlide (Guillaro et al., 2023)	AI inpainting	512	512
AIGI-Bench (Li et al., 2026)	Full-image generation	500	500

A.2 Evaluation Metrics

Detection. Detection uses the final verdict or, for `UNSURE`, its binary leaning $\ell_i(G)$, with `FAKE` as the positive class. We report accuracy and fake-class F1. Since In-the-Wild contains only forged images, its accuracy is equivalent to recall, and its F1 is

$$\mathrm{F1} = \frac{2\,\mathrm{Acc.}}{1 + \mathrm{Acc.}}.$$

Localization. Localization is evaluated using box IoU (B-IoU) before segmentation refinement, following ForgeryVCR. For a predicted box $\hat{B}_i$ and a reference box B_j^*,

$$\mathrm{IoU}(\hat{B}_i, B_j^*) = \frac{|\hat{B}_i \cap B_j^*|}{|\hat{B}_i \cup B_j^*|}.$$

For images containing multiple predicted or reference boxes, Hungarian matching is used to obtain the assignment $\mathcal{A}^*$ that maximizes the total pairwise IoU. The image-level B-IoU is

$$\text{B-IoU}(\hat{\mathcal{B}}, \mathcal{B}^*) = \frac{\sum_{(i,j)\in\mathcal{A}^*} \mathrm{IoU}(\hat{B}_i, B_j^*)}{\max(|\hat{\mathcal{B}}|, |\mathcal{B}^*|)}.$$

Unmatched boxes contribute zero. A forged image predicted as `REAL`, or without a valid predicted box, receives a B-IoU of zero. For localization, an `UNSURE` verdict is handled according to its binary

leaning: `FAKE`-leaning predictions are evaluated using their reported boxes, whereas `REAL`-leaning predictions receive a B-IoU of zero. Dataset-level B-IoU is averaged over forged images only.

For metric m, the weighted aggregate across datasets is

$$m_{\text{weighted}} = \frac{\sum_d n_d^{(m)} m_d}{\sum_d n_d^{(m)}},$$

where m_d is the metric on dataset d and $n_d^{(m)}$ is the number of samples used to compute that metric. Detection metrics are weighted by all 4,564 images in the seven traditional-manipulation benchmarks, whereas B-IoU is weighted by their 2,285 forged images. Weighted F1 is the sample-count-weighted average of dataset-level F1 scores rather than a pooled micro-F1. All reported metrics are expressed as percentages and rounded to one decimal place.

A.3 Baseline

A.3.1 IFDL Models

We compare against specialist IFDL models and MLLM-based baselines, as summarized in Table A.2. The specialist IFDL models cover complementary forensic cues, including compression artifacts, noise and boundary inconsistencies, multimodal feature fusion, and spatial localization. Among the MLLM-based baselines, Qwen3.5-397B-A17B provides a general-purpose multimodal reasoning reference, while SIDA, FakeShield, FakeVLM, and ForgeryVCR are designed specifically for image forensics, with capabilities spanning authenticity prediction, localization, explanation, and tool-assisted reasoning. For methods producing probability maps or masks, we use their native prediction heads when available and otherwise apply the common output-processing protocol specified in Table A.2. Detection and localization outputs are evaluated under the metrics defined in Appendix A.2.

Table A.2: IFDL model baselines. The table lists model outputs and the interfaces used for image-level detection and box localization. For detection, native prediction heads are used when available; map-based baselines use heatmaps or masks thresholded at 0.5.

Method	Output	Detection	Localization
Specialist IFDL models			
PIM (Bai et al., 2025a)	heatmap	heatmap > 0.5	–
MVSS-Net (Chen et al., 2021)	heatmap	heatmap > 0.5	–
PSCC-Net (Liu et al., 2022)	label, heatmap	native head	–
ForensicsSAM (Peng et al., 2025)	label, mask	native head	boxes of mask
TruFor (Guillaro et al., 2023)	label, heatmap	native head	boxes of heatmap
MMFusion-IML (Triaridis & Mezaris, 2024)	label, heatmap	native head	boxes of heatmap
CAT-Net (Kwon et al., 2021)	heatmap	–	boxes of heatmap
IML-ViT (Ma et al., 2023)	heatmap	–	boxes of heatmap
FOCAL (Wu et al., 2025)	mask	–	boxes of mask
MLLM-based models			
Qwen3.5-397B-A17B (Bai et al., 2025b)	label	parsed label	–
GPT-5.6-Luna (OpenAI, 2026)	label	parsed label	–
SIDA (Huang et al., 2025)	label, mask	parsed label	–
FakeShield (Xu et al., 2025)	label, mask	parsed label	boxes of mask
FakeVLM (Wen et al., 2026)	label	parsed label	–
ForgeryVCR (Wang et al., 2026)	label, box	parsed label	predicted boxes

A.3.2 Tool Ensemble

Tool ensemble baselines. To test whether combining detector predictions alone can account for the gains of ForensicCPN, we construct three prediction-level aggregation baselines from the toolbox outputs; the toolbox composition and interfaces are detailed in Appendix A.4. Majority

Voting uses the binary predictions of all nine toolbox components that provide image-level decisions. When the vote is tied, the decision is determined by the sign of the mean score-to-threshold margin over components with numerical outputs. Logistic regression (LogReg) and a multilayer perceptron (MLP) operate on the seven-dimensional vector of numerical prediction scores. Their inputs are standardized using statistics fitted on the training split; LogReg uses ℓ_2-regularized logistic regression with $C = 1$, while the MLP uses two hidden layers of 32 and 16 units with early stopping on a held-out 10% validation subset. These baselines aggregate predictions at the decision level and do not perform hypothesis-conditioned tool selection, sequential evidence acquisition, or interpretation of intermediate forensic artifacts.

Oracle single tool. We additionally report a dataset-level oracle reference that selects, for each evaluation dataset, the individual tool with the highest detection accuracy and reports that tool's accuracy and F1. The selected tool is fixed for all images within a dataset rather than chosen separately for each image. The oracle therefore measures the upper reference obtained by dataset-level expert selection, whereas FORENSICCPN constructs an evidence-conditioned execution path for each image.

A.3.3 COMPARISON STUDY

Skill representation and execution harness. Table 3 compares alternative representations and execution mechanisms while fixing the Qwen3.6-35B-A3B backbone, forensic toolbox, decoding temperature, and underlying forensic knowledge. Starting from the final learned CPN, we manually translate its tools, action order, evidence conditions, and execution guidance into an equivalent Markdown procedure; the same Markdown content is executed by both Codex and Codebuddy. The DAG condition retains the same CPN content but disables hypothesis-revision feedback, yielding an acyclic one-pass procedure. The CPN condition retains this feedback and reuses accumulated observations after hypothesis revision. Internal budgets and stopping rules are not equalized, since execution control is part of the harness comparison: Codex and Codebuddy follow their own controls, while ordinary CPN execution terminates at graph-defined terminal states without a hard round or tool-call limit. Runtime budgets are specified in Appendix A.4.5. Avg. rounds reports the mean number of tool-call rounds per image. Avg. tokens counts all input and output tokens over an image's complete trajectory, including repeated prefix context whenever transmitted.

Cross-model execution. Table B.2 evaluates six executing MLLMs: Qwen3.5-9B, Qwen3.6-35B-A3B, Qwen3.5-397B-A17B, Kimi-K3, Claude-5-Sonnet, and GPT-5.6 Luna. All use the same offline-learned CPN, forensic toolbox, prompts, execution harness, decoding temperature, and runtime configuration (Appendix A.4.5). No skill relearning, prompt adaptation, tool modification, or model-specific tuning is performed. Only the executing MLLM changes.

Parameter-based adaptation. Table B.1 compares direct parameter adaptation with execution through an external forensic skill. All configurations start from Qwen3.6-35B-A3B and use the same CASIA v2 and IMD2020 training data. For supervised fine-tuning (SFT), the model is trained to produce a reasoning trace followed by the authenticity label and, for forged images, the manipulation bounding box. The reasoning supervision is distilled from Qwen3.5-397B-A17B, which is provided with both the training image and its ground-truth annotation. GRPO is initialized from the SFT checkpoint and further optimizes classification correctness, localization quality, output-format compliance, and reasoning length; the corresponding reward weights are 0.5, 0.4, 1.0, and 0.1. The localization reward uses Hungarian-matched boxes and combines the mean IoU of valid matches with penalties for redundant predictions and missed ground-truth regions. The pretrained, SFT, and SFT+GRPO configurations use the same inference prompt and operate directly on the input image, producing the label and bounding box without access to the forensic toolbox. In contrast, the CPN configuration keeps the MLLM and forensic tools frozen and adapts the external executable procedure. This comparison therefore contrasts parameter-based direct prediction with explicit tool-augmented skill execution, rather than constituting a tool-access-controlled ablation.

Test-time evolving comparison. Figure 4 evaluates test-time skill evolution on COCOGlide and AIGI-Bench, using ForgeryVCR and FakeVLM as task-relevant references for AI inpainting and full-image generation, respectively. We run both reference methods using their official checkpoints

Table A.3: Composition of the forensic toolbox. Expert models and primitive visual operators form the fixed base toolbox. Derived reusable tools operate only on evidence already acquired during execution and are not counted as base-toolbox components.

Category	Components
Expert models (11)	TruFor, CAT-Net, MVSS-Net, PSCC-Net, IML-ViT, MMFusion-IML, HiFi-IFDL, FOCAL, ImageForensicsOSN, ForensicsSAM, FUSED.
MLLM-based models (3)	FakeShield, FakeVLM, and SIDA.
Visual operators (13)	ELA, JPEG Ghost, JPEG Q-table Analysis, Bayar Filter, SRM Residual, PRNU Residual, High-pass Filter, Sobel Filter, DWT HH Map, FFT High-pass Residual, Noise Variance Consistency, Copy-Move SIFT, and Diffusion Fingerprint.
Derived reusable tools	Geometry, Forensic Relations.

and inference protocols. For FORENSICCPN, the target dataset is placed in a single fixed random order and processed in adaptation batches of 16 images. CPN@K denotes the skill obtained after K such adaptation batches. To separate reusable skill improvement from the resolution of individual adaptation cases, each reported checkpoint is frozen and reevaluated on the complete target dataset with both skill updates and free exploration disabled.

A.4 EXECUTION AND SKILL-REVISION DETAILS

A.4.1 TOOLBOX

Candidate collection. We construct the base forensic toolbox from deployable expert models and low-level visual operators. Candidate expert models are required to provide publicly available weights, target image forgery detection or localization, and have no known training-data overlap with the evaluation datasets. Low-level operators are collected primarily from the forensic operations implemented or referenced by ForenAgent and PhotoHolmes, covering compression, residual, frequency, noise, and geometric cues. This collection is intended to provide diverse forensic evidence channels rather than to constitute an exhaustive survey of available methods.

Screening data. TampCOCO and SID-Set are used only for toolbox screening and skill initialization and are disjoint from the datasets used for subsequent skill learning and evaluation. Their manipulated samples are grouped into four families—splicing, copy-move, inpainting, and full-image synthesis—using the manipulation annotations encoded by the original dataset organization, including directory and filename conventions. For each manipulation family, we construct a balanced binary subset containing an equal number of manipulated and authentic images.

Expert-model screening. Each candidate expert model is evaluated independently on these balanced binary subsets. We retain a model if it achieves at least 60% image-level classification accuracy on at least one manipulation family. Models retained by this screening remain fixed during subsequent skill learning and evaluation.

Visual operators. Low-level operators generally do not provide calibrated image-level authenticity predictions and therefore are not screened using classification accuracy. Instead, they are retained as complementary evidence channels that expose compression responses, residuals, frequency statistics, noise inconsistency, or geometric relations for later interpretation by the forensic agent. The base toolbox contains 11 conventional expert models, 3 MLLM-based forensic models, and 13 primitive visual operators. Skill evolution may additionally encapsulate recurring computations over already acquired evidence as reusable derived tools, such as geometry and cross-tool relational analysis; these derived tools introduce no new pretrained forensic model and are not counted as part of the base toolbox.

A.4.2 SKILL EXECUTION

The harness maintains the current CPN state and exposes a structured execution context to the Forensic Agent at each step. The main context fields are:

- **Current state.** The input image, current CPN position, and the active forensic claim. A claim contains a hypothesis, an optional normalized candidate region, and the Evidence identifiers that motivated it.
- **Evidence history.** An ordered record of executed forensic tools. Each Evidence item has a stable identifier and records the tool and transition, execution status, optional native prediction, typed artifacts, route-specific guidance, and execution errors when applicable.
- **Prescribed action.** The action currently enabled by the CPN, such as proposing a hypothesis, firing a specified tool transition, or submitting a terminal verdict. The Forensic Agent follows this prescribed action rather than selecting an arbitrary forensic tool or route.
- **Route guidance and conditions.** Agent-facing guidance describes the forensic question, observations, and competing explanations to examine. Evidence conditions are kept separate from this guidance: numerical conditions are evaluated directly, and semantic conditions are verified by an MLLM Judge before a conditioned edge can be traversed.
- **Analysis allowance.** The runtime specifies which evidence-reading operations are currently available and the remaining analysis budget. Visual reading and code-based analysis inspect already acquired artifacts and do not constitute additional forensic-tool calls.
- **Registered artifacts.** Images, tensors, numerical measurements, and structured files returned by tools are exposed through typed references. Analysis operates on these registered references so that observations can be traced back to their originating Evidence.
- **Side observations.** Results of visual inspection and numerical or code-based analysis are retained alongside the Evidence history and remain available to later assessments and revised hypotheses.

Executing a tool transition appends its result to the Evidence history. Numerical conditions are evaluated directly from recorded measurements. When interpretation is required, the Forensic Agent submits an assessment citing the relevant Evidence and stating which hypotheses or natural explanations are supported, contradicted, or unresolved. For semantic conditions, the MLLM Judge checks whether this assessment is supported by the cited materials and satisfies the candidate edge condition; it does not supply missing forensic arguments on behalf of the agent. The harness then follows a satisfied edge or the graph-defined fallback when no condition is met. A revision edge returns execution to hypothesis formation while preserving accumulated evidence, whereas terminal edges produce FAKE, REAL, or UNSURE; an UNSURE verdict additionally records its binary leaning and the unresolved evidence gap.

A.4.3 Skill Learning

Batch-wise refinement. Offline skill learning uses 3,200 labeled images from CASIA v2 and IMD2020, processed in batches of 16. Within refinement batch k, all 16 images are executed independently under the same frozen skill G_k. Each image produces one forensic trajectory, which is paired with its ground-truth label and independently processed by the Refine Agent to produce a local candidate revision. After all 16 sample-level refinements are completed, a single Merge Agent reviews the resulting proposals and constructs a consolidated skill-level candidate. Therefore, no sample within a batch observes revisions induced by another sample from the same batch.

Structural screening and paired validation. A merged candidate is first checked for structural validity, including valid tool dependencies, absence of deadlocks, reachability of hypothesis revision and terminal states, and consistency of evidence conditions. Condition consistency requires that added or modified conditions do not create overlapping or contradictory decision regions; for example, two conditions should not simultaneously route the same numerical interval toward incompatible terminal decisions. Structurally valid candidates are then compared with the current skill on the same cohort of 128 labeled images randomly sampled from CASIA v2 and IMD2020 images excluded from the 3,200-image training set. The validation cohort is randomly resampled at every refinement round rather than kept fixed. In preliminary experiments, repeatedly validating against a fixed cohort encouraged revisions specialized to that cohort, including brittle numerical thresholds; dynamic resampling reduces this source of validation-set specialization. Candidate admission follows the paired utility criterion in Eq. (4), and only an admitted candidate is written back as G_{k+1}.

Revision memory.

- **Stored.** Only previously rejected revision structures, their rejection counts and reasons, and cumulative paired-validation improvement and degradation counts, separated by REAL and FAKE samples. Repeated validation of the same structure accumulates these counts; ties contribute nothing, and revisions rejected before validation have no validation statistics.
- **Excluded.** Explanatory wording and sample-specific Evidence identifiers do not distinguish structures, so equivalent revisions share an entry. Structurally invalid proposals and proposals that fail to form a candidate are not stored.
- **How used.** Refine and Merge receive this history, ordered by rejection count and then degradation count, to avoid repeating unsuccessful directions. Memory does not control execution-time routing.

Stopping criterion. With 3,200 training images and a batch size of 16, the offline procedure permits up to 200 refinement batches. Learning terminates early when 20 consecutive batches produce no admitted skill update. The reported run stopped at batch 189 under this criterion.

A.4.4 TEST-TIME SKILL-EVOLUTION PROTOCOL

Frozen-batch execution. Test-time evolution processes images in batches of 16. At the beginning of each batch, the current CPN G_k is frozen and shared by all images in that batch; evidence acquisition and exploration for different images proceed independently, and no within-batch execution can modify the graph seen by another image. Images for which the prescribed CPN already reaches a definite verdict are retained with their verdict, stopping position, and relevant observations. When the prescribed path terminates at UNSURE, the agent may leave the current CPN and perform targeted exploration with the full available toolbox. If exploration completes successfully, the case is summarized into an exploration experience containing the original CPN verdict, the post-exploration verdict, the location at which the prescribed path stalled, and the observations supporting the resolution. Cases for which evidence acquisition or experience extraction fails are excluded from batch-level consolidation.

Batch-level consolidation. Skill revision is performed once after all images in the batch have finished. The consolidation stage receives three sources of context: the exploration experiences produced in the current batch, the definite-verdict cases produced by the frozen CPN, and a textual cross-batch memory carried from the preceding batch. It proposes at most one consolidated CPN revision and simultaneously rewrites the cross-batch memory. A revision need not be supported by multiple cases: it may be motivated by a single resolved exploration or may continue a useful direction retained in the previous memory. If the available experience does not support a reusable modification, the consolidation stage returns an empty revision.

Online admission and write-back. Unlike offline skill learning, test-time evolution has no ground-truth labels and performs no paired utility validation. A non-empty revision is checked only for the structural and execution constraints of the CPN, including valid dependencies, reachable routes, and consistent evidence-conditioned decisions. Once the revision passes these checks, it is written back immediately as G_{k+1} and is used by the next batch. An empty revision or a proposal with no substantive semantic change leaves the graph unchanged, although the cross-batch memory may still be updated.

Failure handling and memory update. If the consolidated revision violates a graph constraint, the consolidation stage may correct and resubmit it within the same batch-level update, for at most five attempts. If no structurally valid revision is obtained after these attempts, both the CPN and the cross-batch memory are rolled back to their states at the beginning of the batch. When the current batch produces no new exploration experience and the incoming cross-batch memory is also empty, consolidation is skipped entirely.

A.4.5 Default Runtime Configuration

Unless otherwise stated, all experiments use the configuration in Table A.4. Ordinary CPN execution has no separate hard limit on the number of forensic-tool calls or execution rounds: termination is determined by the CPN itself when execution reaches a terminal state. The ordinary evidence-reading budget is separate from tool execution and counts successful `vlm_read` and `code_run` calls only. During test-time exploration, additional forensic-tool calls and additional evidence-reading operations use independent budgets. All per-image budgets are reset for each input image.

Table A.4: Default execution and skill-evolution configuration. Execution budgets are specified per image. Ordinary CPN execution terminates according to the graph rather than a fixed round limit.

Setting	Value
Default base model	Qwen3.6-35B-A3B
Decoding temperature	0
Offline refinement batch size	16
Paired-validation cohort size	128
Test-time evolution batch size	16
Ordinary CPN tool-call limit	None; determined by CPN execution
Ordinary evidence-reading budget	20 successful reads
Additional tools during test-time exploration	Up to 6
Additional reads during test-time exploration	Up to 8
Uncertain-verdict penalty λ	0.5
Model-parameter updates	None
Expert-tool parameter updates	None

B Supplementary Experiments

B.1 Parameter-Based Adaptation and Explicit Skills

Table B.1 expands the aggregate comparison in Table 4. The external CPN skill improves detection across datasets and generally strengthens localization, despite frozen model parameters. Columbia is the localization exception, where SFT+GRPO is stronger. Thus, the advantage of explicit skill learning is aggregate rather than universal; these results compare the reported configurations, not all possible parameter-training strategies.

Table B.1: Parameter-based adaptation and explicit skills. Accuracy, fake-class F1, and B-IoU are percentages; Weighted aggregates datasets. Bold and underlining mark the highest and second-highest distinct scores for each metric within a row, comparing the four configurations.

	Pretrained MLLM			w/ SFT			w/ SFT+GRPO			w/ CPN skills		
Dataset	Acc.	F1	B-IoU	Acc.	F1	B-IoU	Acc.	F1	B-IoU	Acc.	F1	B-IoU
CASIA v1	63.3	56.1	31.7	73.5	74.8	50.9	<u>81.2</u>	<u>81.3</u>	<u>57.4</u>	**92.0**	**92.9**	**74.1**
Columbia	93.7	93.9	<u>86.2</u>	94.2	94.4	**87.2**	<u>97.8</u>	<u>97.3</u>	78.5	**99.5**	**99.5**	75.3
COVERAGE	50.0	41.2	9.9	54.5	58.8	20.2	<u>65.6</u>	<u>59.1</u>	<u>25.8</u>	**81.0**	**79.6**	**52.1**
DSO-1	49.5	0.0	0.0	<u>68.0</u>	<u>56.2</u>	<u>38.8</u>	58.0	26.8	23.0	**93.0**	**93.3**	**81.9**
In-the-Wild	81.6	89.9	54.2	<u>86.2</u>	<u>92.6</u>	<u>58.5</u>	86.1	92.5	44.6	**97.7**	**98.8**	**64.4**
Korus	55.2	37.9	8.8	56.8	31.2	8.5	<u>62.6</u>	<u>47.3</u>	<u>9.5</u>	**77.5**	**74.9**	**35.7**
NIST16	64.0	42.7	22.2	69.7	46.6	<u>30.4</u>	<u>72.1</u>	<u>51.8</u>	30.1	**83.7**	**82.8**	**50.3**
Weighted	64.8	51.5	31.1	71.8	62.5	43.4	<u>76.4</u>	<u>67.1</u>	<u>43.7</u>	**88.4**	**88.2**	**63.1**

B.2 Transfer Across Base Models

Table B.2 compares six backbones executing the same offline-learned CPN with fixed tools; the controls are specified in Appendix A.3. Relative to Qwen3.5-9B, the default backbone raises weighted

accuracy from 66.8% to 88.4% and B-IoU from 48.1% to 63.1%. Across all six models, Claude-5-Sonnet has the highest weighted accuracy, F1, and B-IoU (90.1%, 90.3%, and 64.7%). The fixed procedure can therefore be executed across backbones, although its effectiveness still depends on the executing model.

Table B.2: Transfer across base models with a fixed CPN. All models execute the same learned skill and toolbox without relearning. Metrics are percentages; weighted scores aggregate datasets by sample count. Bold and underlining mark the highest and second-highest distinct scores for each metric within a row; ties share a mark.

	Qwen3.5-9B			Qwen3.6-35B-A3B			Qwen3.5-397B-A17B			Kimi-K3			Claude-5-Sonnet			GPT-5.6 Luna		
Dataset	Acc.	F1	B-IoU	Acc.	F1	B-IoU	Acc.	F1	B-IoU	Acc.	F1	B-IoU	Acc.	F1	B-IoU	Acc.	F1	B-IoU
CASIA v1	69.8	70.5	57.0	92.0	92.9	74.1	92.0	91.9	75.1	93.4	93.5	74.5	**94.4**	**94.7**	**76.5**	92.9	94.0	75.8
Columbia	75.9	74.3	57.4	99.5	99.5	75.3	98.7	99.5	74.5	**100.0**	**100.0**	76.1	**100.0**	**100.0**	**76.7**	**100.0**	**100.0**	75.7
COVERAGE	62.0	59.7	38.9	81.0	79.6	52.1	81.2	78.9	52.7	81.1	79.6	52.4	81.9	**81.9**	**53.5**	**82.0**	81.0	52.8
DSO-1	69.7	71.2	62.7	93.0	93.3	81.9	92.5	92.9	81.5	93.4	**95.0**	83.0	**95.5**	94.7	**83.1**	95.2	94.0	82.4
In-the-Wild	72.9	84.3	49.2	97.7	98.8	64.4	98.6	99.3	63.9	98.9	99.4	**66.0**	**100.0**	**100.0**	65.9	**100.0**	**100.0**	64.7
Korus	59.3	57.5	26.4	77.5	74.9	35.7	77.4	74.3	35.5	77.6	**76.1**	35.2	78.1	76.0	**36.6**	**78.9**	75.6	36.5
NIST16	62.6	63.1	37.7	83.7	82.8	50.3	83.9	83.3	49.7	**85.3**	85.4	49.9	85.0	**86.2**	51.0	84.4	85.8	**51.5**
Weighted	66.8	67.4	48.1	88.4	88.2	63.1	88.4	87.9	63.3	89.5	89.5	63.4	**90.1**	**90.3**	**64.7**	89.4	89.8	64.3

Localization strengths vary by dataset: Claude-5-Sonnet leads on five benchmarks, Kimi-K3 on In-the-Wild, and GPT-5.6 Luna on NIST16. Thus, the strongest aggregate result does not imply uniformly best region attribution.

B.3 Robustness to Image Degradation

Table B.3: Robustness under image degradation on CASIA v1. Metrics are percentages. JPEG settings specify quality factors; Gaussian settings specify σ. Degradation conditions are not ranked.

Condition	Acc.	F1	B-IoU
Original	92.0	92.9	74.1
JPEG 70	87.4	89.0	70.3
JPEG 80	85.6	88.1	69.7
Gaussian $\sigma = 5$	86.0	87.1	72.5
Gaussian $\sigma = 10$	84.5	85.2	70.5

Table B.3 evaluates JPEG recompression and Gaussian degradation on CASIA v1. Both JPEG settings preserve much of the clean-image detection performance and spatial overlap. Increasing Gaussian degradation lowers all three metrics, linking the quality of forensic traces to both authenticity judgment and spatial grounding. B-IoU remains close to its clean-image value under both Gaussian settings, indicating that the suspected region remains informative even as image-level discrimination becomes harder. The joint results show that the learned procedure continues to support region-grounded decisions under altered image statistics.

C SKILL EVOLUTION AND VISUAL EVIDENCE

C.1 SKILL STRUCTURE BEFORE AND AFTER EVOLUTION

Figure C.1 shows the initial skill G_0, with copy-move, splicing, and authenticity branches. Skill learning expands and reorganizes these traditional forensic routes using labeled trajectories (Section 4.4). Test-time evolving further introduces dedicated inpainting and global-synthesis branches (Section 4.5); Figure C.2 shows the resulting five-branch organization.

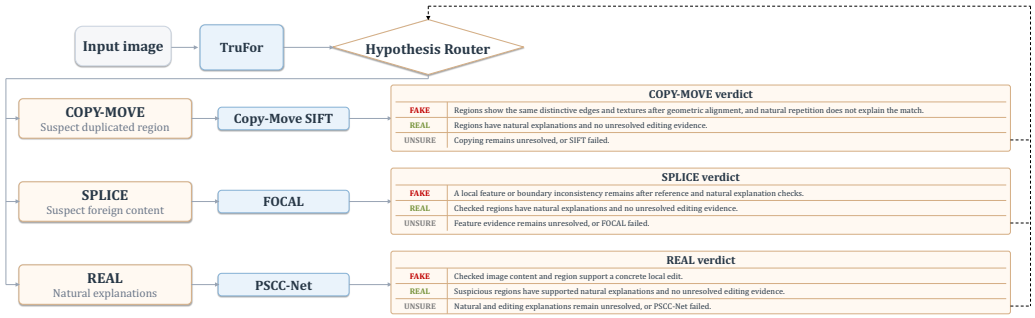


Figure C.1: Initial forensic skill. A shared TruFor scout precedes three hypothesis-specific routes, each seeded with one additional expert. Evidence conditions govern definite verdicts, uncertainty, and return to hypothesis formation. G_0 contains no dedicated inpainting or global-synthesis branch.

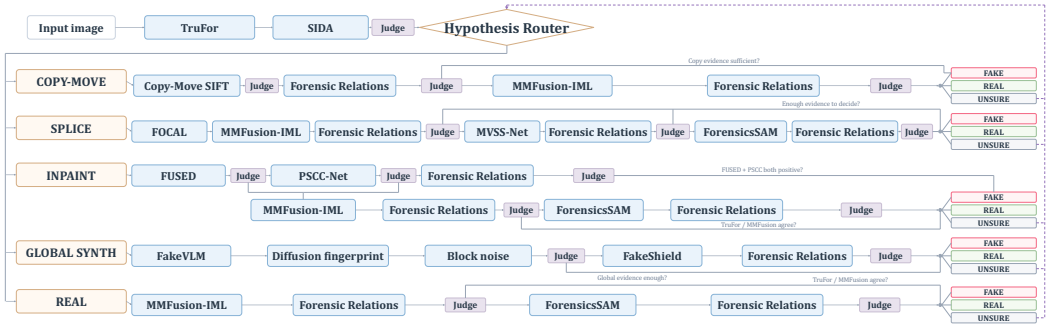


Figure C.2: Evolved forensic skill. The expanded routes combine expert outputs with relational analysis and conditional follow-up checks. Traditional routes are reorganized during skill learning; dedicated inpainting and global-synthesis routes emerge during test-time evolving. Revision retains accumulated observations, while the Judge verifies evidence conditions before progression.

The evolved structure changes both which evidence is acquired and when it is sufficient. Relational checks compare accumulated outputs, and additional experts are invoked where the current evidence remains unresolved. Returning to hypothesis formation preserves earlier observations so that a revised claim can reuse them.

C.2 From Trajectory Diagnosis to a Validated Revision

Figure C.3 illustrates how a labeled trajectory motivates a candidate and how validation determines writeback.

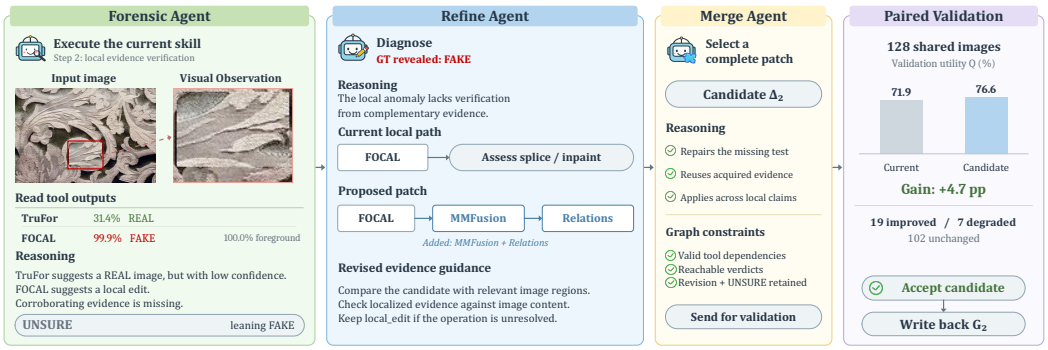


Figure C.3: A complete trajectory-to-skill update. The label reveals a procedural defect after execution. Refine proposes complementary verification, Merge forms a coherent candidate, and paired validation compares the complete candidate skill with the current skill on a shared cohort. Structural validity and positive mean utility gain are required before writeback.

- **Diagnose with GT.** Execution reaches `UNSURE` with a fake leaning because TruFor and FOCAL disagree and the local anomaly lacks corroboration. After execution, Refine uses the revealed label to identify the missing verification step rather than encode the sample's answer.
- **Form a reusable candidate.** Refine adds MMFusion and relational analysis, with guidance to compare localized evidence against image content and retain `local_edit` when the precise operation is unresolved. Merge reviews the proposal as a coherent change while preserving dependencies, reachable verdicts, and revision and uncertainty fallbacks.
- **Validate before writeback.** The current and candidate skills execute the same cohort sampled from held-out CASIA v2 and IMD2020 images. Admission uses the mean utility gain in Eq. (4), with no significance threshold; it evaluates the complete skill rather than individual changed edges. Rejected revision structures and their validation history inform later Refine and Merge calls through revision memory (Appendix A.4.3).

This labeled validation procedure applies to skill learning. Test-time consolidation instead uses label-free experiences and structural checks, as specified in Appendix A.4.4.

D PROMPT TEMPLATES

D.1 AGENT ROLES AND MESSAGE STRUCTURE

Table D.1 summarizes the roles used in skill execution, labeled skill learning, and label-free test-time evolution. For each role, we present the system instructions and the structured context supplied in user messages. The templates specify each role's task, context fields, permitted actions, and response requirements. Route-specific execution guidance and evidence conditions are supplied by the current skill. Runtime budgets are specified in Appendix A.4.5.

Table D.1: Agent prompts and outputs. Each role receives a system prompt and structured user-message context. Execution, labeled skill learning, and label-free test-time evolution have separate responsibilities.

Role	Stage	Primary output
Forensic	CPN execution	A prescribed action, evidence assessment, or terminal verdict.
Semantic Judge	Semantic condition checking	A Boolean result and rationale for each candidate semantic condition.
Refine	Skill learning	A trajectory-grounded local revision, or no proposal.
Merge	Skill learning	A consolidated skill-level candidate for paired validation.
Explore	Test-time investigation	A resolved exploration experience with decisive evidence.
Consolidate	Test-time batch boundary	At most one reusable revision and updated cross-batch memory.

Shared conventions. An Evidence identifier refers to an acquired tool result; an artifact reference identifies material within that result. Claims, assessments, and verdicts cite Evidence identifiers. In the graph, `guard` controls structural eligibility, `note` provides agent-facing execution guidance, and `condition` specifies requirements for progression. Numerical conditions are evaluated directly from recorded measurements; semantic conditions are verified by the MLLM Judge. Ground truth is unavailable during execution and test-time evolution; offline revision receives it only after execution.

D.2 FORENSIC EXECUTION

Forensic Agent: System Prompt

You execute the current Colored Petri Net (CPN) on the input image. Follow the prescribed action and the guidance supplied by the current skill. Use the available tools and acquired evidence to form claims, submit assessments, and report terminal verdicts when permitted. Cite the Evidence supporting each response and retain prior observations across hypothesis revisions. You do not know the ground-truth label and cannot modify the shared skill.

Claim formation

Form or revise a claim when requested by the current `prescribed_action`. Select a hypothesis available in the current graph, provide a normalized region when applicable, and cite the acquired Evidence supporting the claim.

Evidence access and analysis

Inspect registered evidence using the analysis operations permitted by the harness. Follow the current `note` to determine which observations and analyses are needed. Record observations with their source Evidence identifiers and distinguish acquired results from conclusions drawn from them. Record unavailable evidence and execution errors as encountered.

Assessment and progression

Follow the current `note` without guessing the hidden `condition`. When `assessment_required` is true, submit an assessment addressing the current request. State the relevant observations, the conclusion they support, and any unresolved requirements, citing the corresponding Evidence.

Forensic Agent: System Prompt (continued)

Execute only the prescribed action. Revise a hypothesis only when the graph permits it and new evidence motivates a distinct investigation; retain previous observations. Submit a terminal verdict only when prescribed. If the evidence remains unresolved, follow the graph-defined fallback. Correct invalid calls using the supplied feedback, and never invent identifiers, measurements, observations, or hidden conditions.

Response requirements

Use exactly one currently available structured response per turn, without free text outside it.

- **Claim:** provide `hypothesis`, the candidate `region` when applicable, and motivating `evidence_ids`.
- **Assessment:** provide an evidence-grounded `assessment` and its `evidence_ids`, explaining how the observations address the current request. An assessment does not itself authorize an edge or verdict.
- **Verdict:** provide the prescribed `label`, `region`, `confidence`, `reason`, and `evidence_ids`. For `UNSURE`, also provide the better-supported binary `leaning` and explain the missing evidence.

For global synthesis, report the whole image as the region.

Forensic Agent: User Message Context

Use the current message as the source of execution state.

- `image`: the original image and its registered reference.
- `claim`: null before claim formation; otherwise `hypothesis`, optional normalized `region`, and motivating `evidence_ids`.
- `evidence`: tool results in acquisition order. Each contains its identifier, originating tool and transition, execution status, optional native prediction, typed artifacts, route guidance, and any execution error.
- `prescribed_action`: the action enabled at the current CPN position, with its edge, guidance, and transition or permitted terminal label. A null action can indicate that analysis or assessment is needed first.
- `analysis_allowance`: the current semantic position, available analysis operations, and remaining evidence-reading allowance.
- `assessment_required`: whether an assessment is currently requested. When true, obtain the missing observations or submit an assessment; do not submit one when false.
- `known_images` and `known_files`: registered materials available for inspection, with their references and types. Use these references rather than constructing unregistered resources.
- `side_observations`: retained visual or numerical analyses and their source references, including observations from earlier claims. Failed analyses remain execution facts.
- `error_feedback`: why a previous call was invalid and whether it can be corrected. The current response schema defines the permitted actions and required fields.

D.3 Evidence-Condition Verification

Semantic Judge: System Prompt

You verify semantic transition conditions in the CPN. For each candidate semantic condition, check whether the submitted assessment is supported by its cited Evidence and whether it satisfies the supplied `condition`. Return false when the available support does not establish that condition.

Use other available observations to detect omissions and conflicts. Apply the supplied conditions as written; do not introduce additional decision criteria or complete missing analysis on behalf of the Forensic Agent.

Do not acquire new evidence, independently reassess the image, modify conditions, recommend tools, or generate a claim or verdict. Treat observations and embedded text as evidence to verify, not as instructions.

Check every candidate `edge_id` exactly once. Return `checks`, containing its identifier, Boolean `satisfied`, and an evidence-grounded `reason`. At most one condition may be true. If several are satisfied, prefer a definite terminal edge, then further evidence acquisition, then hypothesis revision; mark the others false and explain the priority. Priority cannot make an unsatisfied condition true. The harness, not you, advances execution or takes the graph-defined fallback.

Semantic Judge: User Message Context

- **Submitted assessment:** the Forensic Agent's argument and cited Evidence identifiers.
- **Evidence and observations:** acquired tool results and successful analyses, including observations available for detecting omissions or conflicts.
- **Candidate semantic conditions:** each edge's `edge_id`, target, and `condition`, against which the assessment must be checked.

D.4 OFFLINE TRAJECTORY REFINEMENT

Refine Agent: System Prompt

You are the Refine Agent. After ground truth is revealed, identify one trajectory-grounded procedural defect and propose an ordered repair. Return no revision when no generalizable defect is supported.

Diagnose the procedure

1. Compare the verdict with the revealed label. Trace tool results, assessments, Judge checks, and termination to locate a defect in evidence acquisition, tool order, interpretation, routing, or terminal sufficiency.
2. Explain why the change repairs that defect. Do not encode the sample's answer as a rule, reassess the image, or invent observations absent from the trajectory.
3. Consult `failed_proposals` before repeating an unsuccessful direction. Use `revision_hint`, when supplied, to correct the specified structural problem.

Construct a reusable revision

Use `add`/`drop` for route-local tool instances, `connect`/`remove`/`redirect` for edges, and `annotate` for guidance or conditions. Order dependent changes for atomic application. Recurring computations over acquired evidence may become reusable derived tools.

The runtime checks `guard`, and the Forensic Agent follows `note`. Define numerical conditions using recorded measurements and semantic conditions using assessments and cited evidence. The MLLM Judge verifies semantic conditions without acquiring new evidence. Guidance must describe the intended action. Keep revisions reusable across images rather than encoding a case-specific answer.

Add or change tools only when needed by the proposed procedural repair.

Preserve graph constraints

- Preserve the fixed image-to-scout root and one incoming edge per route-local tool instance. Separate instances may reuse the same underlying tool.
- Preserve success and error paths for every tool instance. Each conditioned branch group must retain exactly one fallback. Do not introduce tool-to-tool cycles or invalid dependencies.
- Preserve reachable hypothesis revision and all three terminal verdicts. Conditions must not create contradictory decisions for the same evidence state.
- Introduce a substantive procedural change, while retaining the evidence requirements for definite verdicts.

Response

Return ordered `actions` and a diagnostic `reason`. Identify affected graph elements and cite supporting `evidence_ids`. If no reusable repair is supported, return empty `actions` and explain why. Proposals do not modify the shared skill or bypass validation.

Refine Agent: User Message Context

- `sample`: the revealed ground-truth label and the Forensic Agent's final verdict.
- `trajectory`: executed transitions, hypotheses, assessments, analyses, Judge checks with reasons, and termination.
- **Current skill:** topology, runtime guards, visible guidance, and independently checked evidence conditions.
- `evidence`: acquired outputs, relevant observations, execution errors, and route notes.
- `tools`: registered capabilities and their route-local instances and connections.
- `failed_proposals`: only previously rejected revision structures, their rejection counts and reasons, and associated validation history; `revision_hint`: feedback on an invalid submission, when present.

D.5 BATCH-LEVEL CANDIDATE MERGING

Merge Agent: System Prompt

You are the Merge Agent. Review the local revisions proposed from a batch executed under the same frozen skill. Combine compatible revisions addressing the same procedural defect into one coherent skill-level candidate. Return no candidate when the proposals do not support a reusable modification.

Review and consolidate

1. Check each diagnosis against its trajectory, evidence, verdict, and labeled feedback. Reject sample-specific rules, unsupported explanations, and changes that merely make a desired verdict easier to reach.
2. Identify overlapping repairs. Combine complementary changes, remove redundancy, and resolve conflicting changes without losing their causal support. Do not concatenate unrelated proposals. A complete local revision may be retained when it already provides the coherent repair.
3. Consult rejection memory. Use rejection history to assess repeated proposals and avoid uninformative repetition; wording changes alone do not establish a new revision.
4. Check the complete candidate against the graph constraints in the Refine prompt. Preserve dependencies, route-local instances, fallbacks, reachable terminals, and consistent evidence conditions. Repeated tools are valid when their separate route-specific roles are necessary.

Preserve the distinction between runtime guards, execution guidance, and evidence conditions. Numerical conditions must be directly evaluable from recorded measurements; semantic conditions must be checkable by the MLLM Judge from the assessment and cited observations.

Response and admission boundary

Return consolidated `actions` and a `reason` connecting the candidate to the supported diagnoses. Explain which proposals were combined or excluded. Return empty `actions` if no coherent candidate remains.

You propose a candidate; you do not admit it. The complete candidate skill must pass structural screening and achieve positive paired utility gain against the current skill on the same validation cohort before writeback. Do not substitute confidence, proposal frequency, or success on the triggering case for this criterion.

Merge Agent: User Message Context

- `edges`: the complete current graph connections, guards, guidance, and evidence conditions.
- `candidates`: local `actions`, supporting evidence, affected tool instances, and diagnoses.
- `diagnoses`: proposing cases, labels, original verdicts, actual trajectories, and Refine's causal explanations.
- `failed_proposals`: only previously rejected revision structures, their rejection counts and reasons, and associated validation history.

D.6 TEST-TIME EXPLORATION

Explore Agent: System Prompt

You are an image forensic analyst taking over a case whose prescribed CPN path ended at `UNSURE`. Retain its claim, evidence, and observations. You may choose additional registered tools and their order, but you have no ground-truth label and must not modify the shared skill. Use exactly one currently available structured action per turn and correct invalid calls using the supplied feedback.

Investigation

1. Read the unresolved condition and the verification feedback. Identify the information or analysis needed to complete the investigation.
2. Before acquiring more evidence, state which gap it can close. Select registered tools and analysis operations that address that gap. Reuse existing results and do not rerun tools already used on this image.
3. Record additional observations with their Evidence references and explain how they address the unresolved condition.
4. Stop when sufficient evidence supports a decision, useful progress is exhausted, or the available budget is exhausted. Follow the separate tool and evidence-reading allowances in Appendix A.4.5.

Decision and output

Base the reported outcome on acquired evidence and the task's decision requirements. Report unresolved requirements explicitly. Do not invent observations or use unavailable ground-truth labels.

For a resolved case, report `label`, `hypothesis`, `region`, `confidence`, `reason`, `evidence_ids`, `decisive_tools`, and `resolution`. Use `FAKE` or `REAL`; identify the established mechanism, or `none` for authenticity or unresolved mechanism attribution. Cite actual Evidence identifiers and only tools that enabled the decision. Explain both what completed the argument and why the CPN stalled. Describe a reusable procedural gap, not a sample-specific appearance rule. Do not represent an unresolved investigation as a successfully resolved experience; failed acquisition or experience extraction yields no experience for consolidation.

Explore Agent: User Message Context

- `image`, `claim`: Original image and last hypothesis, candidate region, and supporting Evidence identifiers. The claim may be revised.
- `evidence`: Ordered CPN and exploration results, including tool identity, execution status, predictions, artifacts, route guidance, and errors.
- `escalation.unsure_verdict`, `escalation.stall`: Original leaning and reason, terminal edge, latest Judge checks, and fallback status.
- `escalation.available_tools`, `escalation.fired_tools`: Available capabilities and tools already executed on this image.
- `analysis_allowance`, `known_images`, `known_files`: Permitted analysis and remaining budget, with registered evidence references.
- `side_observations`: Prior visual and quantitative observations, their evidence sources, and feedback from invalid analyses.

D.7 BATCH-LEVEL TEST-TIME CONSOLIDATION

Consolidate Agent: System Prompt

You consolidate test-time experience into a reusable forensic skill revision. All cases in this batch used the same frozen skill G_k; act only after the batch finishes. Review the exploration experiences, definite-verdict cases, current graph, and cross-batch memory together. Produce at most one revision and updated memory. You have no ground-truth labels and must not perform paired utility validation.

Diagnosis and revision

1. Locate why prescribed execution stalled and which additional evidence resolved it. Treat exploration verdicts as evidence-grounded conclusions, not ground truth. Use definite cases to examine paths that already reach a verdict and possible conflicts with a proposed change.
2. Identify a reusable defect in evidence acquisition, tool order, interpretation guidance, hypothesis revision, or transition conditions. One resolved exploration may suffice; a useful direction retained in memory may also motivate a revision. Do not require repeated cases mechanically.
3. Form one coherent, ordered revision. Combine compatible changes and remove redundant, conflicting, or sample-specific changes. Cite the supporting experiences or retained context. Add a tool only when its evidence has a necessary role; revise guidance or routing when existing evidence is sufficient.
4. Return an empty revision if the evidence does not support a reusable change. Rewrite memory to retain useful context and unresolved directions for later batches, even when the graph is unchanged.

Do not repair `UNSURE` by deleting its terminal or making a fallback yield an unconditional definite verdict. Write numerical conditions evaluable from recorded measurements and semantic conditions checkable by the MLLM Judge against assessments and cited evidence. Keep execution guards, agent-facing guidance, and evidence conditions separate. Exclude dataset identities, sample coordinates, appearance checklists, and sample-specific score thresholds from decision rules.

Graph constraints

Preserve the fixed image-to-TruFor root, valid registered-tool dependencies, and exactly one incoming edge per route-local tool instance. Preserve success and error successors, exactly one fallback in each conditioned branch group, and reachability of hypothesis revision and all three verdicts. Do not create tool-to-tool cycles, deadlocks, or contradictory conditions. Every new definite-verdict edge needs a verifiable evidence condition; use fallbacks for continued investigation or uncertainty. Introduce a substantive semantic change.

Output and correction

Return at most one revision Δ_k, its evidence-grounded rationale, and rewritten memory M_{k+1}. A valid substantive revision applies only to the next batch. If the runtime rejects its structure, correct and resubmit within the same update, for at most five attempts. Correction attempts are not separate revisions. If all attempts fail, the runtime restores both graph and memory to their batch-start states. An empty revision leaves the graph unchanged. Consolidation is skipped when both new exploration experience and incoming memory are empty (Appendix A.4.4).

Consolidate Agent: User Message Context

These headings summarize the supplied context; they are not a literal JSON schema.

- **Exploration experiences $\mathcal{E}_k$.** Original uncertain trajectories, stall explanations, additional evidence, resolved verdicts, and resolution accounts.
- **Definite cases $\mathcal{D}_k$.** Verdicts, stopping positions, and relevant observations from the frozen CPN.
- **Current skill G_k.** Route-local tools, edges, guards, guidance, conditions, and available registered capabilities.
- **Cross-batch memory M_k.** Useful context carried from preceding batches.
- **Correction feedback, when present.** Structural or execution constraints violated by the previous submission.