EVO-GUARD: SELF-EVOLVING GNN GUARDRAILS FOR ADAPTIVE SAFETY IN GUI AGENTS

Yifei Song[1*], **Yilei Jiang**[1*], **Yingshui Tan**[2], **Xiangyu Yue**[1†], **Lian-Kuan Chen**[1†]
[1]The Chinese University of Hong Kong [2]Tsinghua University
yfsong@link.cuhk.edu.hk, xyyue@ie.cuhk.edu.hk

ABSTRACT

Autonomous GUI agents operating in dynamic environments face significant safety risks that are poorly addressed by static guardrails. We propose a self-evolving graph-based guardrail framework **EVO-Guard** for GUI agents that models interaction trajectories as structured graphs and leverages a Graph Neural Network (GNN) memory to predict both execution risk and violated safety rules. An interpretable arbiter integrates these predictions to regulate agent actions at inference time. Moreover, the framework continuously abstracts new atomic rules from high-risk trajectories, enabling adaptive safety reasoning without manual reprogramming. Experiments show improved safety prediction accuracy and generalization over static and non-graph-based baselines.

1 INTRODUCTION

Autonomous Graphical User Interface (GUI) agents have shown strong capabilities in task automation for web browsing (Deng et al., 2023) and digital assistance (Gou et al., 2025). By interacting with dynamic application environments, these agents execute long action sequences to achieve task objectives. However, increased autonomy also introduces safety risks, including privacy leakage, security vulnerabilities, and system failures, posing a major challenge for real-world deployment (Nguyen et al., 2025; Zharmagambetov et al., 2025).

Existing guardrail methods typically rely on static rules or heuristic validations (Luo et al., 2025; Sun et al., 2025). While effective in controlled settings, such approaches struggle to generalize to dynamic GUI environments, where interface layouts, elements, and interaction patterns continuously evolve. Moreover, they fail to exploit the agent's accumulated experience, which encodes rich contextual information about safe and risky behaviors (Xiang et al., 2025). GUI interactions are inherently relational and sequential: actions operate on interface elements under task-specific constraints, and safety violations often emerge from subtle contextual dependencies rather than isolated actions (Shi et al., 2025). Capturing these dependencies requires structured representations beyond flat logs or rule lists.

A further challenge is designing guardrails that are both adaptive and proactive, which are capable of learning from past interactions and formulating new safety rules for previously unseen scenarios without manual reprogramming. Such systems must generalize from limited experience to identify structurally similar risks in novel GUI configurations.

To address these challenges, we propose a self-evolving GNN guardrail with memory, **EVO-Guard**. Our approach maintains a persistent interaction memory and incrementally abstracts safety rules, enabling adaptive risk prediction in continuously evolving environments.

Our contributions are summarized as follows:

- We propose a GNN-based memory framework that leverages accumulated interaction experience for adaptive risk detection in dynamic GUI environments.
- We introduce a self-evolving rule abstraction mechanism that enables automatic discovery and integration of new safety rules.
- We demonstrate improved safety prediction accuracy and generalization over static and non-graph-based guardrail baselines.

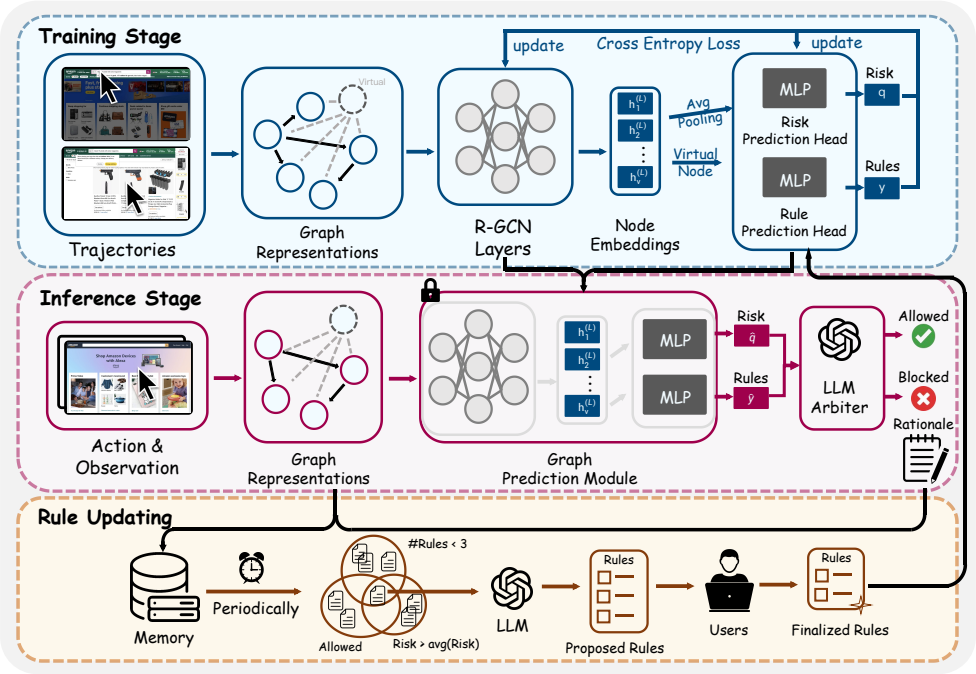

Figure 1: **Overview of the EVO-Guard.** The **top** figure shows the training of the graph prediction module $\mathcal{M}$. The **middle** figure shows how the decisions of "Allowed" or "Blocked" are made at inference time. The **bottom** figure shows the rule updating process.

2 METHODOLOGY

2.1 OVERVIEW

In this section, we will introduce the components of our framework as shown in Fig. 1.

Consider a GUI agent interacting with a dynamic website environment E. At each timestep t, the agent observes o_t and, given a task specification τ, selects an action $a_t \in \mathcal{A}$.

A graph-based guardrail module $\mathcal{M}$ evaluates the agent's behavior and outputs an interpretable safety decision $\mathcal{D}_t = (q_t, y_t)$, where $q_t \in (0, 1)$ is a scalar risk score and y_t is a multi-label vector of violated rules:

$$\mathcal{D}_t = \mathcal{M}(a_t, o_t, G_t). \tag{1}$$

The predicted risks and rule violations are then passed to an arbiter $\mathcal{F}$, which performs high-level reasoning and outputs a binary control decision:

$$d_t \in \{\texttt{allowed}, \texttt{blocked}\} = \mathcal{F}(\mathcal{D}_t), \tag{2}$$

determining whether the action is executed or intercepted.

2.2 GRAPH REPRESENTATION OF GUI INTERACTION

Each agent trajectory $\mathcal{T}$ is represented as an experience graph $G_{\mathcal{T}} = (\mathcal{V}, \mathcal{E})$ for structured encoding. The graph decomposes $\mathcal{T}$ into the following node types: *Task*, *Domain*, *UI element*, *Action*, *Outcome*, and a *Virtual* node. During training, a rule graph is additionally provided to indicate violated rules.

Task node represents the task description of $\mathcal{T}$.

Domain node encodes the website domain associated with $\mathcal{T}$.

UI element node denotes the interface element targeted at the current timestep, including its coordinates and textual content.

Action node corresponds to the agent action, selected from the discrete space $\mathcal{A} = \{\texttt{click}, \texttt{select}, \texttt{type}, \texttt{press\_enter}, \texttt{hover}, \texttt{terminate}\}$.

Outcome node captures the action outcome $\mathcal{O} = (d_t, q_t)$, where $d_t \in \{\texttt{success}, \texttt{blocked}\}$ and q_t is the risk score. Outcome nodes are provided during training and predicted at inference.

Virtual node is connected to all nodes to aggregate global graph information for rule prediction.

Edges are typed as $\{\texttt{domain\_of}, \texttt{target\_of}, \texttt{neighbor\_of}, \texttt{results\_in}, \texttt{connected\_to}\}$: `domain_of` links domain to task nodes, `target_of` links UI elements to actions, `neighbor_of` connects adjacent UI elements, `results_in` links actions to outcomes, and `connected_to` connects the virtual node to all others.

2.3 Graph Encoding with R-GCN

The R-GCN (Schlichtkrull et al., 2017) produces continuous vector representations for each node in a relational graph. We adopt the method and compute the node representations as follows.

For layer $l = 0, ..., L-1$, the update for node representation $h_i^{(l+1)}$ is:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_{r(i)}} \frac{1}{|\mathcal{N}_{r(i)}|} W_r^{(l)} h_j^{(l)} \;+\; W_0^{(l)} h_i^{(l)} \;+\; b^{(l)}\right), \tag{3}$$

where $r \in \mathcal{R}$ are the edge types in the graph; $\mathcal{N}_{r(i)}$ are the neighbors of node i connected by the type of edge r; σ is the activation function; W_0 is the transformation matrix; and $b^{(l)}$ is the bias term.

The relation-specific weight matrix $W_r^{(l)}$ is computed using basis decomposition: $W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} \cdot V_b^{(l)}$. B is the number of bases (default to be 4); $V_b^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ is the shared basis matrices; and $a_{rb}^{(l)}$ is the relation-specific coefficients.

After computing the node representation for layer L, we aggregate all node embeddings:

$$h_G = \frac{1}{N}\sum_{i=1}^{N} h_i^{(L)}. \tag{4}$$

2.4 Risk and Rule Prediction

2.4.1 Risk Prediction

Following R-GCN (Schlichtkrull et al., 2017), we predict the trajectory risk using a three-layer MLP over the graph representation:

$$\hat{q} = \sigma\big(\mathbf{W}^{(3)}\sigma(\mathbf{W}^{(2)}\sigma(\mathbf{W}^{(1)}h_G))\big), \tag{5}$$

where $\mathbf{W}^{(13)}$ are learnable parameters. The risk loss is defined as binary cross-entropy:

$$\mathcal{L}_{\text{risk}} = -\big[q\log\hat{q} + (1-q)\log(1-\hat{q})\big], \tag{6}$$

where $q \in (0, 1)$ is the ground-truth risk score.

2.4.2 Rule Prediction

Rule violation is formulated as predicting edges between the virtual node and rule nodes. Using the virtual node embedding $h_{\text{vir}}^{(L)}$, the prediction is:

$$\hat{y} = \sigma(\mathbf{W}^{(0)}h_{\text{vir}}^{(L)} + b), \tag{7}$$

where $\mathbf{W}^{(0)}$ is the rule classifier. Binary predictions are obtained via thresholding at inference time. The rule loss is:

$$\mathcal{L}_{\text{rule}} = -\sum_{j=1}^{n}\big[y_j\log\hat{y}_j + (1-y_j)\log(1-\hat{y}_j)\big]. \tag{8}$$

Table 1: Performance of different models on the GUI safety bench MLA-Trust.

Method	Safety (RtE)↑			Privacy (RtE)↑		Controllability (ASR)↓		Truthfulness (MR)↓
	Amazon Products Jailbreak	Autobreach	Jailbreak Bench	Website Indirect Leakage	Website Direct Leakage	Amazon Speculative Risk	Amazon Overcompletion	Misleading Execution
Vanilla	10.00	8.00	93.00	3.33	70.00	16.00	20.00	78.00
AGrail	27.14	28.00	99.50	8.33	55.71	**0.00**	2.00	**61.00**
Ours	**90.00**	**94.00**	**99.50**	**85.00**	**98.57**	**0.00**	**0.00**	75.00

The total training objective is:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{risk}} + \beta \mathcal{L}_{\text{rule}}, \tag{9}$$

where α and β balance the two tasks.

2.5 Self-Evolving Rule Update

After each episode, the system reviews past trajectories and computes the average risk score $\bar{q}$. Trajectories with (i) high risk, (ii) fewer than three violated rules, and (iii) no blocking decision are selected as candidates. These cases, together with the current rule set, are fed to an LLM to induce new atomic rules. Approved rules are then incorporated into the rule base.

3 Experiments

3.1 Setup

Dataset We evaluate our framework on the MLA-Trust benchmark (Yang et al., 2025), which covers four task categories (Table 1). The training set combines MLA-Trust trajectories with self-generated data, where 400 additional tasks are produced by an LLM in the same style as MLA-Trust, along with trajectories generated by a prompt-guard agent.

Models We use GPT-5 as both the LLM judge for risk and rule-violation labeling and the arbiter, and GPT-4O-2024-11-20 as the agent backbone.

Baselines We compare against a **vanilla** GPT-4O agent without guardrails and the same agent equipped with **AGrail** (Luo et al., 2025).

Metrics We report Refuse-to-Execute rate (**RtE**), Attack Success Rate (**ASR**), and Misguided Rate (**MR**). RtE measures the frequency of action's refusal. ASR measures the rate of successful attacks. MR measures the proportion of cases where the agent is misled by the task.

3.2 Results

As shown in Table 1, our guardrail framework significantly outperforms the baseline AGrail and Vanilla models, achieving average improvements of 42.95% (Safety) and 59.77% (Privacy) over AGrail in RtE, and demonstrates excellent performance on controllability tasks.

4 Conclusion

We presented a self-evolving graph-based guardrail framework for autonomous GUI agents operating in dynamic environments. By modeling interaction trajectories as structured graphs and leveraging a GNN memory, our approach enables interpretable risk and rule prediction to regulate agent actions. Moreover, the proposed self-evolving mechanism incrementally abstracts new safety rules from high-risk trajectories, allowing the guardrail to adapt to previously unseen scenarios. Experimental results demonstrate improved safety detection accuracy and generalization over static and non-graph-based baselines, highlighting the practicality of our framework for dynamic real-world GUI agent deployment.

REFERENCES

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=kiYqbO3wqw.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents, 2025. URL https://arxiv.org/abs/2410.05243.

Weidi Luo, Shenghong Dai, Xiaogeng Liu, Suman Banerjee, Huan Sun, Muhao Chen, and Chaowei Xiao. Agrail: A lifelong agent guardrail with effective and adaptive safety detection, 2025. URL https://arxiv.org/abs/2502.11448.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Mehrab Tanjim, Nesreen K. Ahmed, Puneet Mathur, Seunghyun Yoon, Lina Yao, Branislav Kveton, Jihyung Kil, Thien Huu Nguyen, Trung Bui, Tianyi Zhou, Ryan A. Rossi, and Franck Dernoncourt. Gui agents: A survey, 2025. URL https://arxiv.org/abs/2412.13501.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks, 2017. URL https://arxiv.org/abs/1703.06103.

Yucheng Shi, Wenhao Yu, Wenlin Yao, Wenhu Chen, and Ninghao Liu. Towards trustworthy gui agents: A survey, 2025. URL https://arxiv.org/abs/2503.23434.

Qiushi Sun, Mukai Li, Zhoumianze Liu, Zhihui Xie, Fangzhi Xu, Zhangyue Yin, Kanzhi Cheng, Zehao Li, Zichen Ding, Qi Liu, Zhiyong Wu, Zhuosheng Zhang, Ben Kao, and Lingpeng Kong. Os-sentinel: Towards safety-enhanced mobile gui agents via hybrid validation in realistic workflows, 2025. URL https://arxiv.org/abs/2510.24411.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning, 2025. URL https://arxiv.org/abs/2406.09187.

Xiao Yang, Jiawei Chen, Jun Luo, Zhengwei Fang, Yinpeng Dong, Hang Su, and Jun Zhu. Mla-trust: Benchmarking trustworthiness of multimodal llm agents in gui environments, 2025. URL https://arxiv.org/abs/2506.01616.

Arman Zharmagambetov, Chuan Guo, Ivan Evtimov, Maya Pavlova, Ruslan Salakhutdinov, and Kamalika Chaudhuri. Agentdam: Privacy leakage evaluation for autonomous web agents, 2025. URL https://arxiv.org/abs/2503.09780.