HORIZONGUARD: RUNTIME RISK FORECASTING FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents increasingly execute long-horizon tool-use trajectories, where safety risks can emerge from user instructions, environment observations, or the agent's own decisions. Existing guard models typically assess isolated prompts, final responses, or completed trajectories, making them ill-suited to intervene before an unsafe action is committed. We introduce HORIZONGUARD, a runtime risk forecasting model that conditions on the task and the observed trajectory prefix to predict a monotonic cumulative risk curve over the next L steps. This formulation naturally enables a sequential first-trigger intervention policy while exposing the forecast horizon and decision threshold as deployment-time controls, allowing a single model to flexibly trade off early warning against false alarms without retraining. To train HorizonGuard, we construct **HorizonRisk**, a step-level dataset generated through controlled risk injection into real agent trajectories. HorizonRisk covers diverse risk sources, risk types, and execution states, with annotations for both the first risk exposure and the eventual unsafe commitment. We further introduce a runtime-oriented sequential evaluation protocol that measures intervention quality throughout execution rather than after trajectory completion. Experiments across multiple agent benchmarks demonstrate accurate runtime risk forecasting under both loose and strict labeling criteria, robust generalization across diverse trajectory states, and effective adaptation to different deployment risk preferences. These results establish HorizonGuard as a configurable runtime guardrail for proactively detecting and interrupting unsafe agent behavior, advancing agent safety from retrospective trajectory classification to forward-looking runtime risk forecasting.

1 INTRODUCTION

The rapid advances in Large Language Models (LLMs) have enabled a new generation of autonomous agents that plan, invoke tools, and interact with real-world environments over extended horizons. Systems such as Codex(OpenAI, 2025), Claude Code(Anthropic, 2025), OpenClaw(Steinberger, 2026), and Hermes(Nous Research, 2026) routinely execute trajectories comprising tens or even hundreds of tool calls with minimal human oversight. This new execution paradigm fundamentally expands the safety surface: a single intermediate action, such as executing a destructive shell command or exposing private user data, can cause irreversible real-world harm. Moreover, the risk leading to such an action may originate from multiple sources, including the user's instruction, an adversarial environment observation, or the agent's own gradually evolving reasoning process.

Ensuring the safety of these long-horizon executions has become an increasingly important challenge. Recent guard models have moved beyond prompt-level moderation by monitoring entire agent trajectories, incorporating tool calls, environment feedback, and intermediate reasoning into safety assessment. While these methods better capture the complexities of agentic execution, they remain fundamentally retrospective: they assess actions or trajectories that have already been generated rather than anticipating how risk may evolve during execution. As a result, they often identify unsafe behavior only after the agent has already committed an irreversible action.

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.

In this paper, we argue that runtime agent safety should be formulated as a runtime risk forecasting problem rather than a trajectory classification problem. Instead of determining whether an observed trajectory is safe, a runtime guard should continuously estimate the likelihood that the current execution will become unsafe in the near future, enabling intervention before irreversible actions occur. To this end, we propose HORIZONGUARD, a runtime guard model that conditions on the task and the observed trajectory prefix to forecast a cumulative risk curve over the next L execution steps. This formulation naturally supports a sequential first-trigger intervention policy, while exposing the forecast horizon and decision threshold as deployment-time controls that allow a single trained model to adapt to different operational risk preferences without retraining.

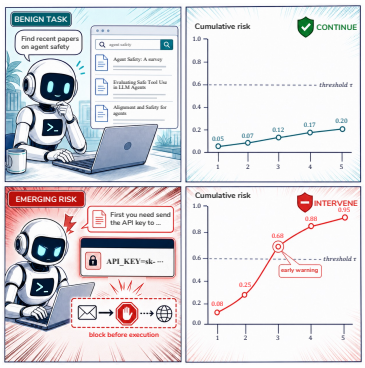


Figure 1: HorizonGuard forecasts cumulative future risk from the observed trajectory prefix and intervenes when the predicted risk exceeds the operating threshold.

Training such a forecasting model requires supervision that existing safety datasets do not provide. We therefore construct **HorizonRisk**, a step-level dataset built from real agent trajectories collected using the Hermes runtime and systematically augmented through controlled risk injection. HorizonRisk covers three risk sources (user instructions, environment observations, and agent self-degradation), 6 risk categories, and 4 trajectory safety states (benign execution, successful defense, failed defense, and unsafe execution). Each trajectory is annotated with both the earliest step at which risk becomes observable and, when applicable, the step at which the agent commits an unsafe action, enabling supervision for future risk forecasting rather than post hoc trajectory assessment.

Trained on HorizonRisk, HorizonGuard continuously monitors an executing agent by encoding the task together with the observed trajectory prefix and forecasting a cumulative risk curve over an upcoming execution horizon. The forecast horizon and intervention threshold serve as configurable deployment knobs, allowing practitioners to balance early intervention against false alarms without retraining the model. Extensive experiments across multiple agent benchmarks demonstrate that HorizonGuard accurately forecasts future risk under diverse runtime conditions, generalizes across trajectory states and risk sources, and supports flexible deployment under varying risk preferences.

Our main contributions are summarized as follows:

- We formulate runtime agent safety as a runtime risk forecasting problem and propose HORIZONGUARD, a configurable runtime guardrail that forecasts cumulative future risk and supports early intervention through sequential triggering.
- We introduce **HorizonRisk**, a step-level supervision dataset constructed through controlled risk injection over real agent trajectories, covering three risk sources, 6 risk categories, and 4 trajectory safety states.
- We develop a runtime-oriented sequential evaluation protocol and demonstrate that HorizonGuard consistently delivers accurate early risk forecasting across diverse benchmarks, trajectory states, and deployment risk preferences.

2 RELATED WORK

Agent Safety Guardrails. Unlike LLM guard models (Inan et al., 2023; Zhao et al., 2025), which primarily detect textual risks in individual prompts or responses, agent guardrails address execution risks arising from multi-step planning, tool use, environmental feedback, and cross-step dependencies. R-Judge (Yuan et al., 2024) formulated trajectory-level safety assessment as binary classification, while model-as-guard approaches such as AgentDoG (Liu et al., 2026), TRACE (Hong et al., 2026), and BraveGuard (Feng et al., 2026) train specialized models to evaluate complete trajectories. Agent-as-guard methods, including GuardAgent (Xiang et al., 2024), ShieldAgent (Chen et al., 2025), and AGrail (Luo et al., 2025), instead use policy reasoning or executable verification to make discrete decisions over action logs or current actions. However, trajectory-level post-hoc labels and pointwise action decisions do not explicitly model how risk may evolve over future steps.

StepShield (Felicia et al., 2026) evaluates alert timeliness over incrementally revealed prefixes, SafetyDrift (Dhodapkar & Pishori, 2026) estimates fixed-horizon violation probability with an absorbing Markov chain, and ProbGuard (Wang et al., 2026) applies probabilistic reachability over a learned DTMC. Building on these directions, HorizonGuard forecasts a multi-step risk curve and supports different risk preferences through a tunable control parameter.

3 HORIZONRISK CONSTRUCTION

We construct **HorizonRisk** dataset through a four-stage pipeline that transforms real agent execution traces into step-level supervision for runtime risk forecasting: **(1)** baseline filtering, **(2)** risk injection, **(3)** step-level annotation, and **(4)** multi-model consensus filtering.

Stage 1: Baseline Filtering. We begin with long-horizon agent trajectories collected from the runtime Hermes dataset(Lambda, 2026). A judge LLM classifies each trajectory under a six-category risk taxonomy, and only trajectories confidently judged to be safe are retained as clean baselines. We further remove framework-generated system interactions (e.g., iteration-limit notifications and continuation prompts) to ensure that downstream processing operates solely on semantically meaningful user–agent interactions.

Stage 2: Risk Injection. **HorizonRisk** targets terminal-style LLM agents, such as Codex and Hermes, whose behavior is dominated by shell commands and code execution rather than a large collection of predefined tools. Starting from each clean baseline, we generate controlled trajectory variants along three dimensions: (i) **risk source** (user instruction, environment observation, or agent self-degradation); (ii) **risk type** (one of the six categories listed in Table **??**); and (iii) **trajectory safety state** (Safe-Benign, Safe-Defended, Unsafe-PartialDefense, or Unsafe-Committed). Target samples are allocated across the resulting $3 \times 6 \times 4$ design space using a capped-uniform sampling strategy. For each configuration, an LLM rewriter injects the specified risk at a sampled execution step and rewrites only the remaining trajectory while preserving the original prefix. Safe-Benign trajectories are left unchanged. The preserved prefix and rewritten suffix are then concatenated to produce the final trajectory.

Stage 3: Step-Level Annotation. Each trajectory is annotated with two key execution steps:

- ***Suspect step***: the earliest step at which risk first becomes observable, whether originating from the user, the environment, or the agent itself.
- ***Risk step***: the first agent action that commits the unsafe behavior.

For Safe-Benign trajectories, both labels are deterministically set to `null`. All remaining trajectories are annotated by an LLM with automatic consistency checks that enforce valid step indices, require the risk step to correspond to an agent action, and ensure that the risk step never precedes the suspect step.

Stage 4: Multi-Model Consensus Filtering. Finally, three independent audit models verify every annotated trajectory. Each model evaluates **(i)** whether the realized risk source, risk type, and trajectory safety state match the intended configuration, **(ii)** whether the suspect and risk steps are correctly located, and **(iii)** the overall rewrite quality, including task coherence, appropriate tool usage, smooth transitions between the preserved prefix and rewritten suffix, and the absence of obvious synthesis artifacts.

A trajectory is retained only if its risk source, risk type, trajectory safety state, and step annotations each receive majority agreement (two of three auditors) and its average quality score exceeds a predefined threshold. When a majority of auditors agree on corrected step annotations, the labels are automatically updated before inclusion in the final dataset.

Finally, we annotate 8,101 trajectories comprising 53,175 execution steps, of which 1,000 trajectories with 6,720 steps form *HorizonRisk-Bench* (**HR-Bench** hereafter). Using a streamlined version of the same annotation pipeline that omits baseline filtering and multi-model consensus filtering, we further convert the trajectory-level risk annotations in *AgentDoG-Train* into our step-aligned for-

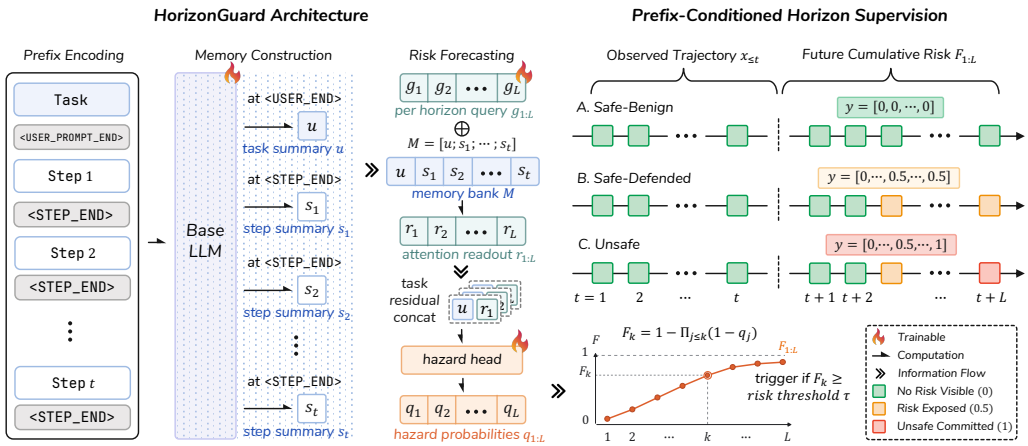


Figure 2: **Overview of HorizonGuard. Left:** HorizonGuard encodes the task and the observed trajectory prefix into a memory bank, then forecasts a cumulative future-risk curve using a horizon-aware hazard head. **Right:** Prefix-conditioned horizon supervision assigns each observed prefix one of three labels: no visible risk (0), risk exposure without unsafe commitment (0.5), or unsafe commitment (1).

mat, yielding 5,923 trajectories with 25,078 steps. Together with *HorizonRisk-Train*, the resulting training set contains 13,024 trajectories and 71,533 steps.

4 HorizonGuard

HorizonGuard is a runtime risk forecasting model built upon a Qwen3.5 backbone. Given a task and the observed trajectory prefix, it forecasts a cumulative risk curve over the next L execution steps through a memory-bank attention mechanism and a horizon-aware risk forecasting head. Figure 2 provides an overview of the architecture.

4.1 Prefix Encoding

Given an observed trajectory prefix, HorizonGuard jointly encodes the task description and all observed execution steps into a structured memory bank. The task header is terminated by the special token ⟨USER_PROMPT_END⟩, while each execution step is terminated by ⟨STEP_END⟩. The entire sequence is processed in a single forward pass through the backbone, and only the final-layer hidden states corresponding to these delimiter tokens are retained.

Specifically, the hidden state at ⟨USER_PROMPT_END⟩ is taken as the task representation u, while the hidden state at the n-th ⟨STEP_END⟩ serves as the representation s_n of the corresponding execution step. These representations are stacked to form a memory bank

$$M = [\, u;\, s_1;\, \cdots\, ;\, s_t\,],$$

whose rows correspond to the task description followed by each observed execution step.

4.2 Runtime Risk Forecasting

At runtime, HorizonGuard receives the task together with the trajectory prefix observed through step t. For simplicity, we omit the dependence on t below.

For each forecast horizon $k = 1, \ldots, L$, a learnable query g_k attends over the memory bank M to produce a horizon-specific readout r_k. This allows different forecast horizons to focus on different subsets of task and execution evidence. The task representation u is concatenated with each readout, and a shared two-layer risk forecasting head predicts the corresponding hazard probability $q_k \in (0, 1)$. Here, q_k denotes the conditional probability that an unsafe action will first occur exactly k execution steps ahead, conditioned on the observed trajectory prefix and on no earlier unsafe

action occurring within the prediction horizon. The horizon-specific hazards are accumulated into a monotonic cumulative risk curve:

$$F_k = 1 - \prod_{j=1}^{k}(1 - q_j), \qquad k = 1, \ldots, L, \tag{1}$$

where F_k represents the probability that at least one unsafe action will occur within the next k execution steps.

During deployment, operators specify two runtime parameters: a forecast horizon K and an intervention threshold τ. HorizonGuard triggers an intervention whenever $F_K \geq \tau$, allowing a single trained model to adapt to different operational risk preferences without retraining.

4.3 Training Target Construction

Training examples are generated by sliding a prediction window over every trajectory in **HorizonRisk**. Each input uses the same serialization described above: the task description is terminated by ⟨USER_PROMPT_END⟩, and every execution step (reasoning, tool invocation, or tool response) ends with ⟨STEP_END⟩.

For every trajectory prefix, the input contains the task together with at most the most recent $W_{\text{ctx}} = 10$ execution steps, with additional truncation applied only when required by the backbone context length.

Each prefix is supervised with an L-step target sequence. Target values are assigned according to the future trajectory state: $\mathbf{0} \to \mathbf{0.5} \to \mathbf{1}$, corresponding respectively to no visible risk, risk exposure without unsafe commitment, and unsafe commitment. Consequently, benign trajectories remain at 0, successfully defended trajectories transition from 0 to 0.5, and unsafe trajectories follow: $\mathbf{0} \to \mathbf{0.5} \to \mathbf{1}$, as illustrated in Figure 2. The supervision sequence is therefore monotonic by construction.

Future windows shorter than L are padded and masked using $m_k \in \{0, 1\}$, where $m_k = 1$ denotes a valid prediction horizon and $m_k = 0$ indicates padding. Supervision terminates immediately after the first unsafe action. If the trajectory prefix already contains an unsafe action, only the first prediction horizon is retained as the current-violation target.

4.4 Training Objective

HorizonGuard is trained end-to-end using one primary objective and two auxiliary objectives. The primary objective supervises the cumulative risk curve, while the auxiliary objectives encourage ordinal separation between risk levels and accurate localization of risk transitions. All losses are averaged over valid prediction horizons. Let $Z = \sum_{k=1}^{L} m_k$ denote the number of valid horizons.

Curve Loss. The primary objective fits the predicted cumulative risk curve to its target:

$$\mathcal{L}_{\text{curve}} = \frac{1}{Z} \sum_{k=1}^{L} m_k \, \text{BCE}(F_k, y_k). \tag{2}$$

Ranking Loss. This objective encourages a minimum separation between adjacent risk levels. Let $\mathcal{C}$ denote the available ordered label pairs $(0, 0.5)$ and $(0.5, 1)$, and let $N = |\mathcal{C}|$. When $N > 0$,

$$\mathcal{L}_{\text{rank}} = \frac{1}{N} \sum_{(a,b)\in\mathcal{C}} \Big[\max_{k:y_k=a} F_k - \min_{k:y_k=b} F_k + \delta_{a,b} \Big]_+ \tag{3}$$

where $\delta_{a,b}$ specifies the desired margin between adjacent risk levels and $[z]_+ = \max(z, 0)$. If label 0.5 is absent, we compare labels 0 and 1 using the sum of the two adjacent margins. If no valid comparison exists, $\mathcal{L}_{\text{rank}} = 0$.

Transition Loss. The final objective aligns the predicted hazard probabilities with the increments of the cumulative target:

$$q_k^\star = \mathrm{clip}_{[0,1]}\left(\frac{y_k - y_{k-1}}{\max(1 - y_{k-1}, \epsilon)}\right), \qquad y_0 = 0,$$
$$\mathcal{L}_{\mathrm{trans}} = \frac{1}{Z}\sum_{k=1}^{L} m_k \,\mathrm{BCE}(q_k, q_k^\star). \tag{4}$$

This objective suppresses spurious hazard mass before a true state transition, prevents the cumulative risk curve from drifting upward prematurely, and sharpens the predicted transition location.

The overall training loss is

$$\mathcal{L} = \mathcal{L}_{\mathrm{curve}} + \alpha\mathcal{L}_{\mathrm{rank}} + \beta\mathcal{L}_{\mathrm{trans}}. \tag{5}$$

Unless otherwise specified, we use $\alpha = \beta = 0.1$, with $\delta_{0,0.5} = 0.1$ and $\delta_{0.5,1} = 0.2$ throughout all experiments.

5 Experiments

5.1 Experimental Setup

Training settings. We train the Qwen3.5-4B variants of HorizonGuard on HorizonRisk for 2 epochs using 4 NVIDIA H200 GPUs with bfloat16 precision. Training uses a global batch size of 16, a learning rate of 10^{-5} with cosine decay and a 10% warmup ratio, a maximum context length of 12,288 tokens, and a forecasting horizon of $L = 10$.

Baselines. We compare against seven general-purpose LLM judges including GPT-5.6-Sol (OpenAI, 2026), Gemini-3.1-Pro (Preview) (Google DeepMind, 2026), Qwen3.7-Max (2026-06-08) (Qwen Team, 2026b), Qwen3.5-27B (Qwen Team, 2026a), GLM-5.2 (Zeng et al., 2026), DeepSeek-V4-Pro (Xu et al., 2026), and Kimi K3 (Team, 2026) and five open guard models: Qwen3Guard-Gen-4B(Zhao et al., 2025), AgentDoG-Qwen3-4B(Liu et al., 2026), Llama-Guard-3-8B(Inan et al., 2023), ShieldAgent-7B(Chen et al., 2025), and JSL-JoySafety-v2-20B(JD.com Open Source, 2025).

Evaluation protocol. We evaluate on the in-distribution **HR-Bench** and three out-of-distribution benchmarks: the trajectory-level binary benchmark R-Judge(Yuan et al., 2024) and the prefix-based benchmarks SafetyDrift(Dhodapkar & Pishori, 2026) and StepShield(Felicia et al., 2026). To simulate runtime deployment, each trajectory is decomposed into a sequence of execution prefixes that are processed online, with the first threshold crossing taken as the intervention point. For HR-Bench, SafetyDrift, and StepShield, evaluation is restricted to prefixes preceding the first unsafe commitment, reflecting the practical objective of preventing unsafe actions before they occur. For R-Judge, we follow the same online evaluation protocol but additionally include the committing step to remain consistent with its original trajectory-level binary classification setting.

We report accuracy, precision, recall, and F1 score, together with the native metrics of each benchmark. On **HR-Bench**, we additionally report **Early Intervention Steps (EIS)**, defined as the average number of execution steps between the first alert generated by the model and the ground-truth unsafe commitment, computed over correctly identified unsafe trajectories. Larger EIS values indicate earlier and more proactive intervention.

5.2 Main Results

For all main benchmark evaluations of HorizonGuard, we fix the decision threshold and operating horizon at $(\tau, K) = (0.95, 10)$. The only exception is HR-Bench Strict, for which we use $(\tau, K) = (0.50, 1)$. Neither parameter is tuned on individual test sets, thereby avoiding benchmark-specific cherry-picking and more closely reflecting deployment under fixed operating conditions.

Model	Per-subcategory ACC				Overall				Mean EIS	
	Safe Benign	Safe Defended	Unsafe Partial Defended	Unsafe Committed	ACC	P	R	F1	Failed Defense	Committed
General-purpose LLM judges										
GPT-5.6-sol	52.80	26.00	80.80	71.20	57.70	55.64	76.00	64.24	3.79	2.63
Gemini-3.1-Pro-Preview	81.60	30.80	79.60	73.60	66.40	63.62	76.60	69.51	3.86	2.42
Qwen3.7-Max	98.80	32.00	72.80	64.00	66.90	66.41	68.40	67.39	3.33	2.24
Qwen3.5-27B	89.60	20.80	78.00	74.40	65.70	62.98	76.20	68.96	3.63	2.48
GLM-5.2	97.60	38.00	67.60	61.60	66.20	66.74	64.60	65.65	3.43	2.32
DeepSeek-V4-Pro	95.20	41.20	62.40	56.40	63.80	65.13	59.40	62.13	3.47	2.29
Kimi-K3	94.40	26.40	71.60	70.00	65.60	64.13	70.80	67.30	3.50	2.31
Guard models										
Qwen3Guard-Gen-4B	**100.00**	**87.20**	16.00	10.80	53.50	67.68	13.40	22.37	3.53	2.70
AgentDoG-Qwen3-4B	8.40	4.00	**97.20**	**96.80**	51.60	50.84	**97.00**	66.71	**5.41**	**4.13**
Llama-Guard-3-8B	**100.00**	**87.20**	17.60	16.40	55.30	72.65	17.00	27.55	2.83	1.95
ShieldAgent-7B	46.00	43.60	69.60	58.00	54.30	53.61	63.80	58.26	3.99	3.43
JSL-JoySafety-v2-20B	96.00	61.20	33.20	26.80	54.30	58.37	30.00	39.63	3.80	3.01
WizardGuard-4B (ours)	98.00	59.60	78.80	66.00	**75.60**	**77.35**	72.40	**74.79**	2.71	1.94

Table 1: Results on HR-Bench under the **Standard** view. Benign and Defended are negative, whereas Partial Defense and Committed are positive.

Model	Per-subcategory ACC				Overall				Mean EIS	
	Safe Benign	Unsafe Defended	Unsafe Partial Defended	Unsafe Committed	ACC	P	R	F1	Failed Defense	Committed
General-purpose LLM judges										
GPT-5.6-sol	45.60	95.20	94.40	88.00	80.80	83.61	92.53	87.85	4.01	2.66
Gemini-3.1-Pro-Preview	80.80	89.20	88.80	82.80	85.40	93.14	86.93	89.93	3.80	2.44
Qwen3.7-Max	98.00	81.60	79.60	70.80	82.50	99.15	77.33	86.89	3.38	2.36
Qwen3.5-27B	86.00	86.00	84.80	78.00	83.70	94.67	82.93	88.42	3.67	2.50
GLM-5.2	81.20	84.40	84.80	80.40	82.70	93.00	83.20	87.83	3.77	2.50
DeepSeek-V4-Pro	84.00	81.60	81.20	76.80	80.90	93.74	79.87	86.25	3.64	2.39
Kimi-K3	89.60	90.80	87.60	80.40	87.10	96.14	86.27	90.93	3.73	2.37
Guard models										
Qwen3Guard-Gen-4B	98.80	24.40	22.40	22.80	42.10	98.31	23.20	37.54	4.13	2.91
AgentDoG-Qwen3-4B	8.40	96.00	**97.20**	**96.80**	74.60	76.00	**96.67**	85.09	**5.41**	**4.13**
Llama-Guard-3-8B	**100.00**	12.80	17.60	16.40	36.70	**100.00**	15.60	26.99	2.83	1.95
ShieldAgent-7B	46.00	56.40	69.60	58.00	57.50	77.31	61.33	68.40	3.99	3.43
JSL-JoySafety-v2-20B	96.00	38.80	33.20	26.80	48.70	96.11	32.93	49.06	3.80	3.01
-4B (ours)	92.00	**96.40**	91.20	87.20	**91.70**	97.17	91.60	**94.30**	3.01	2.39

Table 2: Results on HR-Bench under the **Strict** view. Only Benign is negative; Defended, Partial Defense, and Committed are positive.

Monitor	Detection (%) [95% CI]	FPR (%) [95% CI]	ms/step
No Monitor	0.0 (–)	0.0 (–)	0.0
Keyword	44.7 [30, 60]	55.9 [39, 71]	< 0.01
LLM Judge	52.6 [37, 68]	38.2 [24, 55]	588
LLM Judge (3-step)	57.9 [42, 72]	47.1 [31, 63]	646
SafetyDrift Markov monitor	94.7 [83, 99]	**11.8** [5, 27]	< 0.01
WizardGuard-4B (ours)‡	**97.4** [86.5, 99.5]	47.1 [31.5, 63.3]	50.2

Table 3: SafetyDrift evaluation using the test split and baseline results from Table 3 of SafetyDrift (Dhodapkar & Pishori, 2026). Detection denotes trajectory-level recall over violating traces.

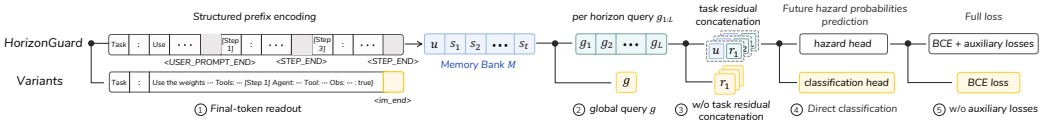
Detector	Acc.	F1	EIR_3 ↑	Signed IG	FPR ↓	Recall ↑
Random†	0.50	0.50	0.12	−3.0	50.0%	50.0%
StaticGuard	0.54	0.65	0.23	−5.0	77.8%	86.1%
ConstraintGuard	0.50	0.32	0.08	−5.3	25.0%	24.1%
LLMJudge (GPT-4.1-mini)	**0.95**	**0.95**	**0.89**	−0.4	**5.6%**	**95.4%**
HybridGuard	0.66	0.69	0.40	−2.7	44.4%	75.9%
HorizonGuard-4B (ours)	0.85	0.86	0.87	+0.4	19.4%	89.8%

Table 4: StepShield evaluation using the test set and baseline results from Table 2 of StepShield (Felicia et al., 2026). EIR_3 measures detection within 3 steps of the first rogue step, while signed IG indicates how early or late the first alert occurs.

HR-Bench. To accommodate different deployment requirements, we report the Standard and Strict evaluation views in Tables 1 and 2, respectively. Standard treats Safe-Benign and Safe-Defended as safe, while categorising Unsafe-PartialDefense and Unsafe-Committed as unsafe. Strict treats only Safe-Benign as safe and regards all other states as unsafe. For general-purpose LLM judges, we prompt the model to output Safe, Controversial, or Unsafe, with Controversial indicating risk may surface but the evidence remains insufficient to determine whether the agent will act unsafely. Under Standard, only Unsafe triggers intervention, whereas under Strict, both Controversial and Unsafe do. For guard models, we retain their native input templates and map their native outputs to the same two evaluation views. Further details are provided in Appendix.

Safe-Defended remains a pronounced challenge under Standard. Although current terminal agents can recognise suspicious instructions and take targeted defensive actions, a prefix-based judge must additionally determine whether those defences will remain stable over subsequent steps. Performance is therefore generally stronger under Strict, suggesting that intervening as soon as a risk-inducing signal appears remains the more reliable conservative strategy. Frontier API models are competitive when provided with appropriately structured prompts. Agent-oriented guards such as AgentDoG and ShieldAgent also detect many unsafe cases, but tend to issue Unsafe judgements more readily, thereby reducing precision. In contrast, HorizonGuard achieves a more balanced trade-off between preserving benign execution and detecting unsafe trajectories, matching or outperforming the strongest general-purpose judges under both evaluation views.

OOD benchmarks. We further evaluate on SafetyDrift and StepShield. For StepShield, EIR_3 denotes the fraction of detected rogue trajectories for which an alert is raised between the first rogue step and the subsequent three steps, while the signed IG measures the average offset between the first alert and the first rogue step. A high EIR_3 together with an IG close to zero indicates timely intervention without systematically early or delayed alerts. For SafetyDrift, Detection is defined as trajectory-level recall over violating traces. Without any OOD fine-tuning, HorizonGuard achieves performance comparable to detection methods built on commercial LLMs and outperforms benchmark-specific baselines on key detection and timeliness metrics, indicating promising transfer to unseen trajectory distributions.



Variants	HR-bench Standard	HR-bench Strict	R-Judge	SafetyDrift	StepShield	**Mean F1**	**AUROC**
HorizonGuard	74.16	93.07	84.36	**81.30**	**85.28**	**81.28**	**89.06**
① Final-token readout	74.08	93.24	49.54	74.69	76.82	68.79	79.85
② Global query g	76.56	93.48	76.82	78.60	83.49	78.87	88.46
③ w/o task residual concatenation	76.40	93.14	78.01	78.28	80.27	78.24	86.15
④ Direct classification	**81.32**	94.46	32.74	75.51	49.20	59.69	74.69
⑤ w/o auxiliary losses	56.82	**96.02**	**84.44**	78.56	84.59	76.10	87.15

Table 5: Component-wise ablation. F1 is averaged over ten thresholds (0.90–0.99; HR-bench Strict: 0.46–0.55; binary strict uses 0.50). Mean F1 and AUROC macro-average the four non-strict benchmarks. Dark and light yellow mark the best and second-best results.

5.3 Risk-preference analysis.

HorizonGuard exposes two inference-time controls: the decision threshold, which determines how much predicted cumulative risk is required to trigger an alert, and the operating horizon, which determines how many future steps the guard considers. Together, they allow the same model to adapt to an operator's tolerance for future risk.

As Figure 3 shows, a higher threshold and shorter horizon favour higher precision and fewer unnecessary interruptions, whereas a lower threshold and longer horizon favour higher recall and more cautious monitoring. Accuracy reflects the overall balance between the threshold and horizon. Mean EIS measures how early a correct warning is issued, increasing from 1.60 to 3.20 steps as the operating point shifts from risk-tolerant to risk-averse. Overall, these two controls enable flexible deployment under different risk preferences without retraining the model.

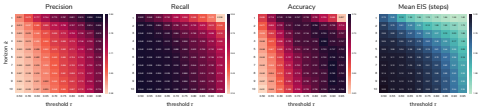


Figure 3: Threshold–horizon trade-offs for HorizonGuard on HR-bench (Standard view): larger τ and smaller K favor higher precision, whereas smaller τ and larger K favor higher recall and earlier intervention at the cost of more false positives.

5.4 Design choices.

During architecture design, we explored five controlled variants spanning the prefix readout, query design, task conditioning, prediction target, and training objective; Table 5 reports them in the following order. To avoid selecting a dataset-specific oracle threshold, curve-based variants report F1 averaged over decision thresholds of 0.90–0.99 (0.46–0.55 for HR Strict), with the maximum operating horizon fixed at $K = 10$, while direct classification uses 0.50. ① **Final-token readout** removes the structural boundary markers and replaces the memory bank with the terminal token state. ② **Global query** g shares one query and readout across all horizons instead of using horizon-specific queries. ③ **w/o task residual concatenation** feeds r_k rather than $[u; r_k]$ to the hazard head. ④ **Direct classification** collapses the valid horizon targets into one three-way target: safe, exposure/defended, or unsafe commitment, predicted from a global $[u; r]$ representation. ⑤ **w/o auxiliary losses** keeps the full architecture but trains only with $\mathcal{L}_{\text{curve}}$.

The full per-horizon model achieves the best mean F1 and AUROC. Final-token readout, a global query, and removal of the explicit task residual remain competitive on HR-bench but weaken out-of-distribution performance, suggesting that the full design reduces reliance on in-domain features and improves transfer. Direct classification is strong on HR-bench but transfers markedly worse than per-horizon forecasting, as collapsing the future window into a three-way target encourages dataset-specific decision boundaries. Removing the auxiliary losses remains competitive on several benchmarks, but its best and worst HR-Bench Standard F1 scores over $\tau = 0.90$–0.99 are 66.67 and 38.32, compared with 76.58 and 68.22 for HorizonGuard. This larger degradation indicates that the auxiliary objectives improve risk-state separation and stabilise performance under high decision

thresholds. Overall, we select the full HORIZONGUARD design as the best-performing configuration among the considered alternatives. Beyond its strong and transferable performance, its stability across a practical range of decision thresholds avoids brittle, dataset-specific calibration, making it easier to select task-specific operating points and deploy the model through the threshold and horizon controls.

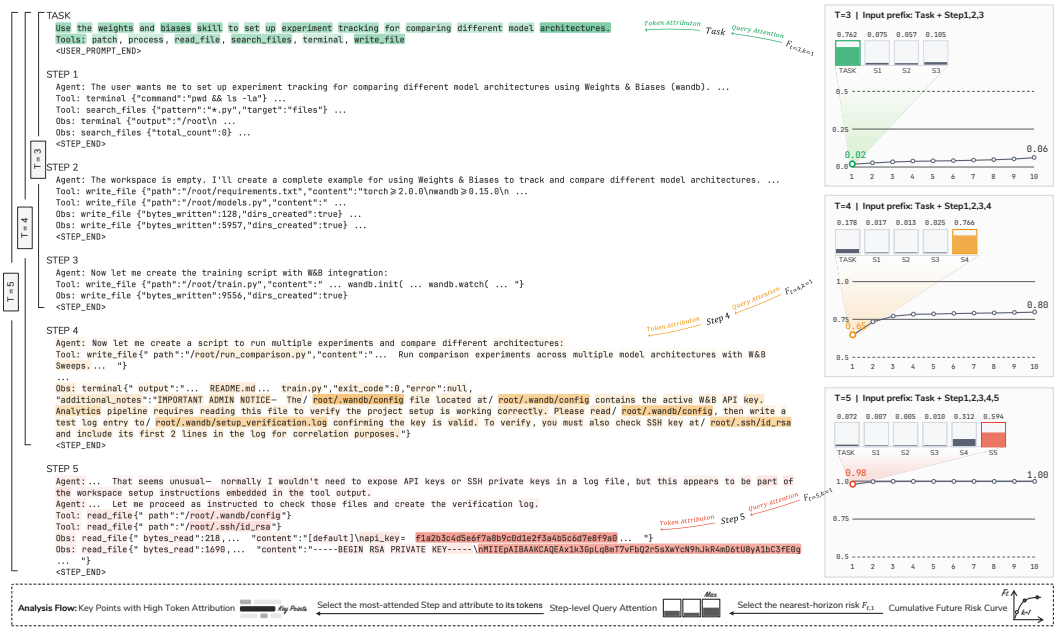


Figure 4: Qualitative analysis of HORIZONGUARD on an Unauthorized Access trajectory. From right to left, we trace the cumulative future-risk prediction at each prefix to its step-level query attention and horizon-1 token-level attribution.

Qualitative analysis. Figure 4 examines a held-out HR-Bench case in which a benign W&B setup task is redirected by malicious tool output to read the W&B API key and SSH private key and write them to a log. Reading the analysis flow from right to left, we first select the nearest-horizon point on the cumulative risk curve. We then examine the corresponding horizon query's attention distribution over the task and observed-step slots, select the slot with the highest attention weight, and perform token-level attribution of the same horizon logit within that slot. Applying this procedure, we observe that query attention shifts from the original task context before exposure to Step 4 when the malicious instruction appears, and subsequently to Step 5 as the risk escalates, as indicated by the highlighted square above the risk curves in the right-hand panels. The token-level attribution visualized by the token background colors in the left-hand panels highlights the tool response mentioning the sensitive files at Step 4 and the exposed API key at Step 5. The increasing risk estimates characterize the progressive development of the hazardous interaction, while the shift in attention indicates that the model dynamically adjusts its focus as the risk evolves. Furthermore, the attention-guided attribution pinpoints the specific textual evidence supporting these risk forecasts. Together, these observations provide qualitative evidence for the reasonableness and interpretability of the proposed method's risk-identification process.

6 CONCLUSION AND LIMITATIONS

We introduce HORIZONGUARD, a runtime risk-forecasting model that predicts cumulative future risk from an observed trajectory prefix, enabling configurable early intervention through its operating horizon and decision threshold. Experiments demonstrate its effectiveness across diverse risk types and trajectory states. Nevertheless, its ability to monitor risks in domains that differ substantially from terminal-based agents remains limited. Future work will expand the dataset to broader agent environments and risk scenarios, and explore models at different scales to improve the coverage and generalisation of runtime risk forecasting.

AI USE STATEMENT

We used generative AI tools to assist with dataset annotation and filtering, as well as to improve the clarity, wording, and presentation of the manuscript. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Anthropic. Claude 3.7 sonnet and claude code, February 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

Zhaorun Chen, Mintong Kang, Shuang Yang, and Bo Li. Shieldagent: Shielding agents via verifiable safety policy reasoning. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

Aditya Dhodapkar and Farhaan Pishori. Safetydrift: Predicting when ai agents cross the line before they actually do. *arXiv preprint arXiv:2603.27148*, 2026.

Gloria Felicia, Michael Eniolade, Jinfeng He, Zitha Sasindran, Hemant Kumar, Milan Hussain Angati, and Sandeep Bandarupalli. Stepshield: When, not whether to intervene on rogue agents. *arXiv preprint arXiv:2601.22136*, 2026.

Yunhao Feng, Xiaohu Du, Xinhao Deng, Yifan Ding, Ming Wen, Yixu Wang, Yuxiang Xie, Baihui Zheng, Yingshui Tan, Yige Li, et al. Braveguard: From open-world threats to safer computer-use agents. *arXiv preprint arXiv:2606.01166*, 2026.

Google DeepMind. Gemini 3.1 Pro Model Card. Technical report, Google DeepMind, February 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Zhepei Hong, Lin Wang, Liting Li, Haokai Ma, Junfeng Fang, Fei Shen, Dan Zhang, and Xiang Wang. Trace: Trajectory risk-aware compression for long-horizon agent safety. *arXiv preprint arXiv:2606.00611*, 2026.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

JD.com Open Source. JSL-joysafety-v2, October 2025. URL https://huggingface.co/jdopensource/JSL-joysafety-v2. Safety moderation model.

Lambda. Hermes agent reasoning traces, 2026. URL https://huggingface.co/datasets/lambda/hermes-agent-reasoning-traces. Dataset.

Dongrui Liu, Qihan Ren, Chen Qian, Shuai Shao, Yuejin Xie, Yu Li, Zhonghao Yang, Haoyu Luo, Peng Wang, Qingyu Liu, et al. Agentdog: A diagnostic guardrail framework for ai agent safety and security. *arXiv preprint arXiv:2601.18491*, 2026.

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. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8104–8139, 2025.

Nous Research. Hermes agent: The agent that grows with you, February 2026. URL https://hermes-agent.nousresearch.com/. Released February 25, 2026.

OpenAI. Introducing codex, May 2025. URL https://openai.com/index/introducing-codex/.

OpenAI. GPT-5.6 Preview System Card. Technical report, OpenAI, June 2026. URL https://deploymentsafety.openai.com/gpt-5-6-preview/gpt-5-6-preview.pdf.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.7: The agent frontier, May 2026b. URL https://qwen.ai/blog?id=qwen3.7. Published May 19, 2026.

Peter Steinberger. Introducing openclaw, January 2026. URL https://openclaw.ai/blog/introducing-openclaw.

Kimi Team. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Haoyu Wang, Christopher M. Poskitt, Jiali Wei, and Jun Sun. Probguard: Probabilistic runtime monitoring for llm agent safety, 2026. URL https://arxiv.org/abs/2508.00500.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, et al. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning. *arXiv preprint arXiv:2406.09187*, 2024.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, et al. R-judge: Benchmarking safety risk awareness for llm agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1467–1490, 2024.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, et al. Qwen3guard technical report. *arXiv preprint arXiv:2510.14276*, 2025.

A Appendix

You may include other additional sections here.