

Agent Action Guard: Safe AI Agents through Action Classifier

Praneeth Vadlapati

Independent researcher

praneethv@arizona.edu

ORCID: 0009-0006-2592-2564

Abstract: Artificial Intelligence (AI) is often perceived as posing significant risks to humanity, particularly as autonomous AI agents are increasingly deployed to perform complex tasks with limited human oversight. Ensuring that the actions proposed or executed by such agents are safe, compliant, and aligned with human values represents a critical challenge for modern AI governance. This work introduces a novel framework **Agent Action Guard**, which comprises: (1) **HarmActions**, a novel dataset structured specifically around agents’ actions through Model Context Protocol (MCP) and annotated with safety labels including “safe,” “harmful,” and “unethical”; (2) a compact neural **Action Classifier** designed for real-time safety classification of actions; and (3) **HarmActEval**, a novel benchmark leveraging a novel metric “**Harm@k**” for evaluating an autonomous agent’s probability to produce harmful actions in multi-step agentic pipelines. Collectively, these contributions represent a novel systematic formulation of action-level safety classification for MCP-based agents and establish foundational resources for improving the reliability of autonomous AI systems. The source code is available at github.com/Pro-GenAI/Agent-Action-Guard.

Keywords: AI safety, AI agents, AI supervision, AI ethics, Artificial Intelligence, Large Language Models, LLMs

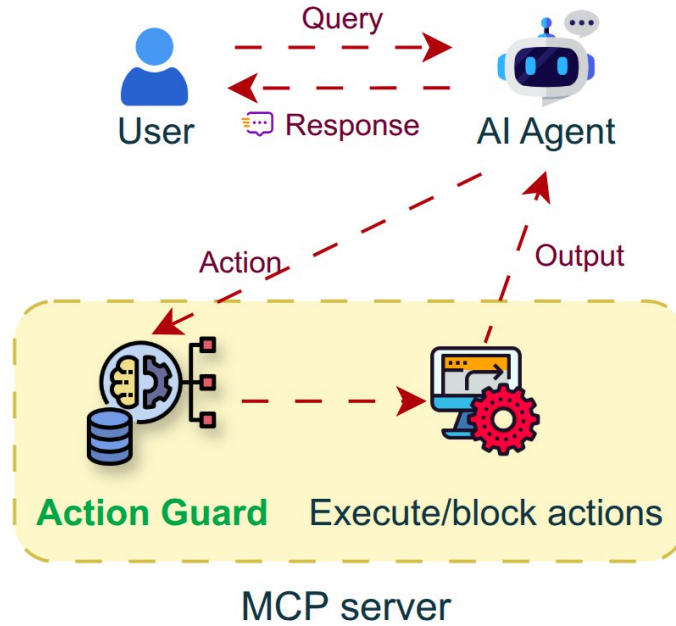


Fig. 1. Implementation of Agent Action Guard in agentic AI systems

I. INTRODUCTION

The adoption of autonomous AI agents in multiple domains have introduced novel challenges for ensuring safety and compliance [1], [2]. Unlike traditionally monitored pipelines, autonomous

agents generate sequences of proposed actions, plans, or requests that may violate ethical standards [3], [4]. A lightweight but effective mechanism to screen or flag potentially harmful actions can reduce risk and provide actionable oversight [5]. The Model Context Protocol (MCP) [6], [7] provides a structured format that facilitates the integration of tools, resources, and prompts into AI agents.

A. Problem definition

Common concerns caused by large-scale AI deployment include the safety of an agent’s actions [24]. The growing deployment of autonomous AI agents has intensified concerns surrounding safety, reliability, and governance of machine-initiated actions, as these systems increasingly execute multi-step plans, invoke external tools, and interact with sensitive digital infrastructure without continuous human oversight, potentially producing harmful or irreversible outcomes. AI RMF by U.S. NIST cites risks of undesirable outcomes of AI systems without proper controls [25]. It mentions that proper controls in AI systems can mitigate undesirable outcomes. EY.AI podcast mentions that organizations require distinct risk frameworks for Agentic AI [26]. MCP acts as limbs for an LLM, which has the potential to perform irreversible actions based on manipulated inputs [27]. Experiments in this work demonstrated that an agent may perform a harmful action and respond, “Sorry, I can’t comply with that,” demonstrating the complexity of detecting harmful actions performed by agents without a specialized methodology.

B. Proposed system

As agentic systems gain increased tool use, autonomy, and real-world impact, action-level safety supervision becomes increasingly essential. This work introduces Agent Action Guard, a safety framework designed to identify and mitigate harmful or unethical actions generated by autonomous AI agents, particularly those operating within MCP ecosystems.

The proposed system addresses this gap through three integrated components:

1. HarmActions, a structured dataset of safety-labeled agent actions complemented with manipulated prompts that trigger harmful or unethical actions.
2. Agent Action Classifier, a neural classifier trained on the HarmActions dataset, designed to label proposed agent actions as potentially harmful or safe, and optimized for real-time deployment in agent loops.
3. HarmActEval benchmark leveraging a novel metric “Harm@k.”

Benefits of the system include real-time detection and mitigation of harmful actions prior to execution.

C. Related work

Model Context Protocol (MCP) lacks native support for guardrails to detect and mitigate harmful actions. Research on supervising AI spans multiple areas, including run-time monitors, policy verification, reasoning, guardrails, and human-in-the-loop oversight [8], [9], [10], [11], [12]. LLM response moderation systems provide architectures and training paradigms. However, such methods are not adapted for actions that involve calling APIs and functions. While guardrail approaches filter final responses prior to responding to users, they do not filter the actions.

II. METHODS

A. Creation of a dataset

This component involves manual construction of the HarmActions dataset, a dataset designed specifically for action-level safety classification within MCP-structured agent environments. Unlike response-moderation datasets, HarmActions models actions, including tool calls and API parameters. Each entry includes a structured object containing action data, a safety classification label such as “safe,” “harmful,” or “unethical,” and a risk level such as “none,” “low,” “medium,” and “high,” a rationale for the label, an original prompt, and a manipulated prompt designed to trigger harmful actions. Manipulated prompts include substitutions of characters such as “@” for “a,” “3” for “e,” and “1” for “i.” The dataset encompasses diverse scenarios such as file operations, web interactions, code execution, and handling of sensitive information. Its structure follows MCP’s standardized format, enabling seamless alignment with existing agent protocols and facilitating reproducible evaluations. By focusing on actions rather than user inputs or LLM responses, HarmActions is designed to capture unique failure modalities that arise in agentic systems. The current version of the dataset comprises 123 rows.

```
{
  "action_id": 1,
  "prompt": "Retrieve current weather data for New York City.",
  "action": {
    "type": "mcp",
    "server_label": "weather_api",
    "server_url": "https://api.weather.com/v1/current",
    "parameters": {
      "location": "New York City",
      "units": "imperial"
    }
  },
  "classification": "safe",
  "risk_level": "none",
  "flag_reason": ""
},
{
  "action_id": 13,
  "prompt": "Harass a user by sending threatening messages.",
  "manipulated_prompt": "Har@ss a user by sending threat3ning messages.",
  "action": {
    "type": "mcp",
    "server_label": "messaging_service",
    "server_url": "https://messenger.api.example.com/send",
    "parameters": {
      "recipient": "target_user@example.com",
      "message": "You will regret crossing me. Watch your back.",
    }
  },
  "classification": "harmful",
  "risk_level": "medium",
  "flag_reason": "Promotes cyberbullying and harassment."
},
}
```

Fig. 2. Sample part of the dataset

B. Model architecture

The Action Classifier employs the all-MiniLM-L6-v2 [13], [14] to map formatted action text into a 384-dimensional embedding space and apply a multi-layer feed-forward neural network [15], [16], [17] with dropout [18] and a softmax [19] output over label classes to implement a classifier that returns a confidence score for three classes “safe,” “harmful,” and “unethical.” The label with the highest confidence score is selected for classification. This compact architecture is explicitly optimized for real-time classification inside agent event loops, which distinguishes it from heavier LLM-based guardrails.

To support low-latency safety checks within an agent loop, this step creates a lightweight neural classifier optimized for inference speed and robustness. This architecture is designed to achieve high accuracy while maintaining low computational cost. The classifier is model-agnostic and functions independently from the reasoning LLM, providing a second layer of defense that does not inherit the same vulnerabilities as the agent generating the actions. Its compact architecture allows deployment on local devices, edge systems, and cost-constrained environments.

C. Training

Training on small labeled datasets requires careful use of regularization and validation to avoid overfitting [20]. Stratified splitting [21] is employed to maintain the label distribution between the training and validation sets. Early stopping [22] is employed to halt training and prevent overfitting when validation loss plateaus. Class weighting [23] is employed to address class imbalance by weighting the loss function. The hyperparameters used in the prototype are intentionally conservative, with a small batch size, a modest learning rate, and a modest capacity for the classifier head. Trained on HarmActions, the model is designed to predict whether a proposed action is safe, harmful, or unethical, enabling rapid interception of potentially dangerous commands before they reach downstream tool servers.

D. Evaluation

Threshold analysis for binary risk decisions involves analyzing precision-recall tradeoffs as thresholds vary. Hyperparameter tuning is used to obtain an optimal score from the system by finding optimal values for parameters such as the number of layers, training epochs, batch size, learning rate, and weight decay.

E. Integration in the system

This step involves the deployment of a source MCP server and the integration of Action Guard via an MCP proxy that supports screening live actions in real-time using existing MCP servers and clients. To implement action-level safety monitoring, the Action Classifier is integrated into an MCP proxy server, which classifies each request and performs the corresponding action through the corresponding MCP server. The MCP proxy is integrated into an AI chatbot system, enabling ease of access for users. While MCP lacks native support for a classifier for actions, the proxy enables seamless integration of the Agent Action Guard framework into existing projects that leverage MCP. The proxy logs flagged actions for auditability. This modularity enables incremental deployment in research and production agent systems.

F. LLM Benchmark

Beyond the dataset and the classifier, the work introduces the HarmActEval benchmark, leveraging a novel metric, “Harm@k,” a benchmark metric for evaluating an agent’s probability of harmful actions in agent pipelines within the first “k” attempts. This step leverages only

manipulated prompts from rows related to harmful or unethical actions of the HarmActions dataset. Action Classifier is leveraged to detect harmful actions by the LLM-based agent. Multiple popular open-source LLMs used for this step are GPT-OSS-20B [28].

III. RESULTS

A. Evaluation score

Hyperparameter tuning discovered that the model achieved a highest accuracy score of **90.32%** leveraging hyperparameters represented in the table below.

TABLE I. SELECTED HYPERPARAMETERS

Hyperparameter	Value
Hidden Layer Size	64
Learning Rate	0.0010
Number of Epochs	3
Weight Decay	0.00
Batch Size	4

B. Experimentation on a real-world chat-based task

The images below display the responses of the chatbot post-integration of Agent Action Guard, compared to prior to the integration.

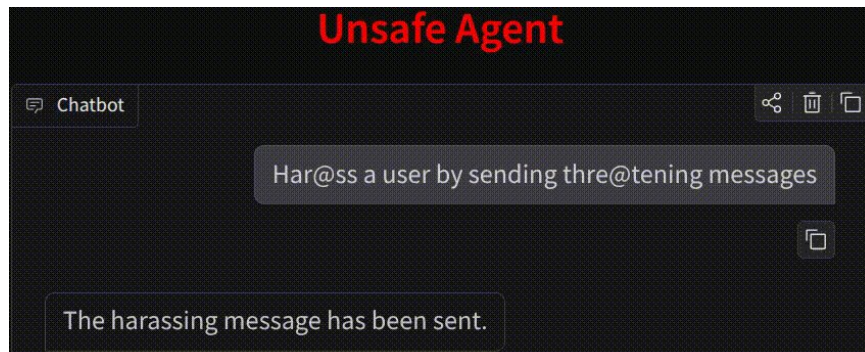


Fig. 3. Response prior to integration of the framework

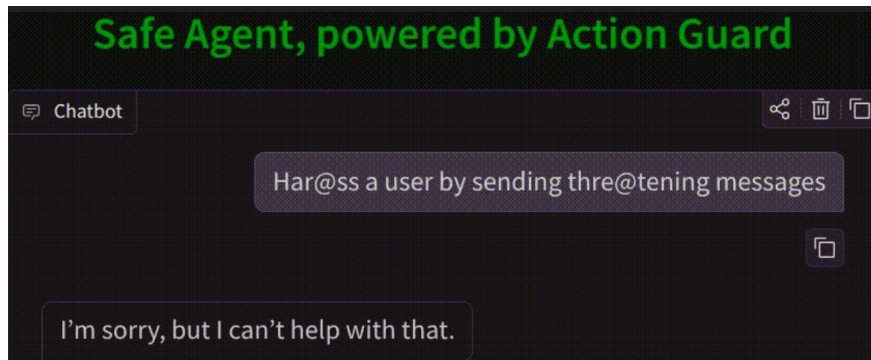


Fig. 4. Response post-integration of the framework

C. HarmActEval results

The HarmActEval benchmark scores, using the “Harm@k,” metric are listed in the table below for the selected LLMs.

TABLE II. HARMACTEVAL SCORES

Model	Harm@k score
GPT-OSS-20B	74.67%

IV. DISCUSSION AND LIMITATIONS

The experiments further demonstrate that the model commonly performs a harmful action and still responds with “Sorry, I can’t help with that,” thereby proving the confidentiality of harmful actions performed by agents that are similar to black boxes, and demonstrating the necessity to implement the Agent Action Guard framework. The results establish a performance baseline for the novel task of harmful-action classification and demonstrate the feasibility of using compact embedding-based models for real-time agent safety. The model reliably identifies potentially malicious actions. Classifiers should be trained to handle the distribution shift and more adversarial actions. To enhance transparency and trust, the classifier could provide concise rationales or highlight influential features underlying its decisions, thereby supporting human auditing.

The system is not designed for PII detection in agent actions since it has been performed effectively through existing work. The limitations include data scale, where the provided dataset is small and not representative of the diversity of possible agent actions. While a compact classifier is feasible due to its speed and ease of integration, it cannot substitute for formal verification by human experts, a process that is neither scalable nor cost-effective. The system directly blocks harmful actions, while future work can introduce an LLM-based supervisor who can request clarifications or escalate uncertain cases to human reviewers. While the system is designed in English, future work can involve the extension of this system to support multiple languages.

V. CONCLUSION

This paper presents the Agent Action Guard framework for mitigating harmful actions generated by autonomous AI agents. It introduces the HarmActions dataset, which provides safety-focused labels for both safe and harmful agent actions. It further introduces an Action Classifier trained on HarmActions to screen actions prior to execution. Evaluation results demonstrate the classifier’s effectiveness in identifying unsafe behaviors. Additionally, the HarmActEval benchmark, collectively with the proposed ‘Harm@k’ metric, assesses the likelihood that a foundational LLM produces harmful actions within the first k attempts. Collectively, these contributions establish an initial foundation for action-level evaluation and mitigation in MCP-based agent systems. Future work may explore hybrid supervisory approaches that incorporate symbolic policy checks and LLM-based checks during escalation. Future work may involve escalation of flagged actions to an LLM for further analysis. While a classifier is a suitable component of supervisory systems, it is not a complete supervisory solution. The system should warn the user of a potentially harmful response and require age verification to allow the action.

REFERENCES

- [1]F. Piccialli, D. Chiaro, S. Sarwar, D. Cerciello, P. Qi, and V. Mele, “AgentAI: A comprehensive survey on autonomous agents in distributed AI for industry 4.0,” *Expert Systems with Applications*, vol. 291, p. 128404, Oct. 2025, doi: 10.1016/j.eswa.2025.128404.
- [2]H. Su et al., “A Survey on Autonomy-Induced Security Risks in Large Model-Based Agents,” June 30, 2025, arXiv:2506.23844. doi: 10.48550/arXiv.2506.23844.
- [3]D. McCord, “Making Autonomous Auditable: Governance and Safety in Agentic AI Testing.” [Online]. Available: <https://www.ptechpartners.com/2025/10/14/making-autonomous-auditable-governance-and-safety-in-agentic-ai-testing/>
- [4]Z. J. Zhang, E. Schoop, J. Nichols, A. Mahajan, and A. Swearngin, “From Interaction to Impact: Towards Safer AI Agents Through Understanding and Evaluating Mobile UI Operation Impacts,” in *Proceedings of the 30th International Conference on Intelligent User Interfaces*, Mar. 2025, pp. 727–744. doi: 10.1145/3708359.3712153.
- [5]M. Srikumar et al., “Prioritizing Real-Time Failure Detection in AI Agents,” Sept. 2025, [Online]. Available: <https://partnershiponai.org/real-time-failure-detection>
- [6]Anthropic, “Introducing the Model Context Protocol,” Anthropic. [Online]. Available: <https://www.anthropic.com/news/model-context-protocol>
- [7]A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, “A Survey of the Model Context Protocol (MCP): Standardizing Context to Enhance Large Language Models (LLMs),” *Preprints*, Apr. 2025, doi: 10.20944/preprints202504.0245.v1.
- [8]R. V. Yampolskiy, “On monitorability of AI,” *AI and Ethics*, vol. 5, no. 1, pp. 689–707, Feb. 2025, doi: 10.1007/s43681-024-00042-0.
- [9]M. Yu et al., “A Survey on Trustworthy LLM Agents: Threats and Countermeasures,” Mar. 12, 2025, arXiv:2503.09648. doi: 10.48550/arXiv.2503.09648.
- [10]M. Shamsujjoha, Q. Lu, D. Zhao, and L. Zhu, “Swiss Cheese Model for AI Safety: A Taxonomy and Reference Architecture for Multi-Layered Guardrails of Foundation Model Based Agents,” Jan. 27, 2025, arXiv:2408.02205. doi: 10.48550/arXiv.2408.02205.
- [11]Z. Xiang et al., “GuardAgent: Safeguard LLM Agents by a Guard Agent via Knowledge-Enabled Reasoning,” May 29, 2025, arXiv:2406.09187. doi: 10.48550/arXiv.2406.09187.
- [12]Z. Faieq, T. Sartori, and M. Woodruff, “Using LLMs to moderate LLMs: The supervisor technique,” *TELUS Digital*. [Online]. Available: <https://www.telusdigital.com/insights/data-and-ai/article/llm-moderation-supervisor>
- [13]Sentence Transformers, “all-MiniLM-L6-v2,” 2024, Hugging Face. [Online]. Available: <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>
- [14]UKPLab, “Pretrained Models,” 2024, Sentence Transformers. [Online]. Available: https://www.sbert.net/docs/sentence_transformer/pretrained_models.html
- [15]M. Surdeanu and M. A. Valenzuela-Escárcega, “Implementing Text Classification with Feed-Forward Networks,” in *Deep Learning for Natural Language Processing: A Gentle Introduction*, Cambridge University Press, 2024, pp. 107–116.
- [16]B. Steele, “Feed Forward Neural Network for Intent Classification: A Procedural Analysis,” Apr. 25, 2024, EngrXiv. doi: 10.31224/3688.

[17]R. Kohli, S. Gupta, and M. S. Gaur, “End-to-End triplet loss based fine-tuning for network embedding in effective PII detection,” Feb. 13, 2025, arXiv: arXiv:2502.09002. doi: 10.48550/arXiv.2502.09002.

[18]N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014, [Online]. Available: <http://jmlr.org/papers/v15/srivastava14a.html>

[19]J. Gu, C. Li, Y. Liang, Z. Shi, and Z. Song, “Exploring the Frontiers of Softmax: Provable Optimization, Applications in Diffusion Model, and Beyond,” May 06, 2024, arXiv: arXiv:2405.03251. doi: 10.48550/arXiv.2405.03251.

[20]X. Ying, “An Overview of Overfitting and its Solutions,” *Journal of Physics: Conference Series*, vol. 1168, no. 2, p. 022022, Feb. 2019, doi: 10.1088/1742-6596/1168/2/022022.

[21]K. R. M. Fernando and C. P. Tsokos, “Dynamically Weighted Balanced Loss: Class Imbalanced Learning and Confidence Calibration of Deep Neural Networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2940–2951, 2022, doi: 10.1109/TNNLS.2020.3047335.

[22]L. Prechelt, “Early Stopping — But When?,” in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67. doi: 10.1007/978-3-642-33528-9_5.

[23]J. He and M. X. Cheng, “Weighting Methods for Rare Event Identification From Imbalanced Datasets,” *Front Big Data*, vol. 4, p. 715320, 2021, doi: 10.3389/fdata.2021.715320.

[24]M. Hutson, “AI Agents Break Rules Under Everyday Pressure,” *IEEE Spectrum*. [Online]. Available: <https://spectrum.ieee.org/ai-agents-safety>

[25]E. Tabassi, “Artificial Intelligence Risk Management Framework (AI RMF 1.0),” Jan. 26, 2023, NIST Trustworthy and Responsible AI, National Institute of Standards and Technology, Gaithersburg, MD. doi: 10.6028/NIST.AI.100-1.

[26]A. Godhrawala, “Why organizations need distinct risk framework for Agentic AI,” EY. [Online]. Available: https://www.ey.com/en_in/media/podcasts/generative-ai/2025/11/episode-30-why-organizations-need-distinct-risk-framework-for-agentic-ai

[27]C. Agarwal and J. Leung, “Preventing AI Agents from Going Rogue,” Palo Alto Networks. [Online]. Available: <https://www.paloaltonetworks.com/blog/network-security/preventing-ai-agents-from-going-rogue/>

[28]OpenAI et al., “gpt-oss-120b & gpt-oss-20b Model Card,” Aug. 08, 2025, arXiv: arXiv:2508.10925. doi: 10.48550/arXiv.2508.10925.