

Multimodal Large Language Agents for Autonomous GUI Understanding and Computer Interaction

Jarl Omar

Department of Computer Science and Engineering, University of Nevada, Reno, Reno, NV,
USA.

jarl.work@unr.edu

Elvira Klotz

School of Computing, Clemson University, Clemson, SC, USA.

elvira@clermson.edu

Fenna Gemen

Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence,
KS, USA.

fenna.gemen@ku.edu

Abstract

Autonomous graphical user interface understanding and computer interaction represent a frontier in artificial intelligence systems research, where multimodal large language agents are expected to perceive screens, interpret interface semantics, plan multi-step tasks, and execute actions across heterogeneous software environments. This paper examines the system-level architecture, infrastructure requirements, governance challenges, and long-term sustainability of such agents. It argues that the central difficulty is not merely visual recognition or language generation, but the integration of perception, memory, planning, tool invocation, action grounding, and accountable execution within dynamic socio-technical infrastructures. The discussion distinguishes desktop, web, and mobile settings, emphasizing trade-offs among latency, privacy, robustness, accessibility, and auditability. It also considers how evaluation benchmarks shape engineering priorities, often privileging short-horizon task success while underrepresenting failure recovery, fairness, and human oversight. Governance concerns include permission boundaries, prompt injection, data leakage, unsafe automation, and the distribution of responsibility among users, vendors, and deployers. The paper concludes that trustworthy multimodal computer-use agents require layered architectures, sandboxed execution, transparent telemetry, continuous evaluation, and policy frameworks that treat automation as an infrastructural commitment rather than a single model capability. By connecting technical design with institutional accountability, the analysis offers a forward-looking agenda for deploying agents that can operate graphical environments without eroding human control, equity, or system resilience.

Keywords

multimodal large language agents, graphical user interfaces, autonomous computer interaction, human-computer interaction, AI governance, system architecture, robustness, fairness, sustainability.

1. Introduction

The evolution of computer interaction has moved from command-line instructions to graphical environments whose meaning is distributed across pixels, text, layout, icons, menus, dialogs, and dynamic state. Autonomous agents that can operate these environments must therefore combine visual perception, linguistic reasoning, procedural memory, and action execution in ways that traditional automation scripts cannot easily replicate. Early reinforcement learning systems demonstrated that sequential decision-making can be learned from interaction, but their dependence on structured action spaces limited transfer to open-ended software [1]. Transformer architectures later provided a scalable substrate for sequence modeling and attention-based representation learning [2], while large language models showed that broad task competence can emerge from pretraining and few-shot conditioning [3]. These developments created the technical conditions for agents that reason over instructions and adapt to novel interfaces, but they did not by themselves solve the problem of grounded computer use.

Multimodal systems extended these foundations by aligning visual and textual representations, allowing models to connect images with language in a shared semantic space [4]. Vision-language models further demonstrated few-shot visual question answering and cross-modal inference [5], and embodied multimodal models showed that language can mediate planning and control in physical or simulated environments [6]. In parallel, language agents began to combine reasoning traces with actions, reflection, and memory, producing iterative problem-solving loops that resemble human trial-and-error behavior [7], [8]. Generative agent simulations illustrated how memory, planning, and social interaction can be orchestrated over long horizons [9], while surveys of autonomous agents mapped the emerging design space of profiling, memory, planning, and action modules [10]. These strands converge in the idea of a multimodal large language agent that observes a graphical user interface, interprets user intent, decides on a sequence of operations, and executes them through keyboard, mouse, touch, or programmatic channels.

The practical promise of such agents is substantial. They could automate repetitive enterprise workflows, assist users with accessibility needs, manage complex software configurations, and provide natural-language control over applications that lack robust APIs. Smartphone-oriented agents have already explored how multimodal models can navigate mobile interfaces through visual understanding and action prediction [11], and recent work on generalist agents for automated computer interaction has advanced integrated perception, planning, and execution in desktop-like settings [12]. However, the same capabilities that make these systems useful also introduce structural risks. An agent with broad screen access can expose sensitive data, trigger irreversible actions, or be manipulated by adversarial interface content. Its behavior depends not only on model quality but also on execution sandboxes, permission systems, logging infrastructure, update pipelines, and organizational policies. This paper therefore treats autonomous GUI understanding as a systems and governance problem, not merely a model benchmark problem.

2. Conceptual Foundations and System Architecture

A multimodal large language agent for computer interaction can be understood as a layered system. The perceptual layer converts screenshots, accessibility trees, document object models, and operating system metadata into representations that the reasoning layer can use. The reasoning layer interprets goals, decomposes tasks, maintains state, and selects actions. The execution layer translates selected actions into low-level events such as clicks, keystrokes, scrolls, and text entry, or into higher-level API calls when available. A memory layer stores

episodic traces, task progress, user preferences, and environmental constraints. A governance layer enforces permissions, monitors behavior, records audit trails, and mediates escalation to humans. These layers are not independent; failures in perception can produce incorrect plans, while failures in governance can transform a benign planning error into a security incident. The architecture must therefore be designed for observability and containment rather than assuming that model accuracy will prevent all harmful outcomes.

The foundation-model substrate shapes what the agent can learn and how flexibly it can generalize. Attention-based architectures enable long-range dependency modeling across interface states and instructions [2], and large-scale pretraining supports transfer to unfamiliar applications [3]. Vision-language alignment enables the agent to map visual elements to semantic labels and user goals [4], while few-shot multimodal learning allows rapid adaptation to new interfaces without extensive retraining [5]. Embodied and robotic multimodal models demonstrate that language can serve as a bridge between high-level intent and low-level control [6]. Yet GUI environments differ from physical environments in important ways: they are discrete, symbolic, highly structured, and often instrumented with accessibility metadata. This makes some tasks easier, because the agent can read text labels and widget roles, but it also creates adversarial surfaces, because interface content can be modified to mislead the agent. The architecture must therefore combine learned perception with deterministic verification where possible.

Agent loops provide a conceptual model for iterative interaction. Reasoning-and-acting frameworks interleave thought, action, and observation, allowing the agent to revise plans based on feedback [7]. Reflection mechanisms enable the agent to critique past attempts and adjust future behavior, improving performance on multi-step tasks [8]. Generative agents show that memory retrieval and planning can sustain coherent behavior over extended periods [9], and surveys of autonomous agents emphasize the importance of modular design, tool use, and environment modeling [10]. In GUI settings, the loop must operate under tight latency constraints because users expect responsive interaction. It must also handle partial observability, since dialogs, pop-ups, and background processes may hide relevant state. Recent multimodal generalist agents for automated computer interaction illustrate how these components can be integrated into a single controller that observes screens and executes actions [12]. The key architectural question is how much reasoning should occur before action, how much state should be persisted, and how responsibility should be divided between the model, the runtime, and the human supervisor.

3. Perception, Tool Use, and Action Grounding

Perception in GUI environments is inherently multimodal. A screenshot provides visual layout, color, iconography, and spatial relationships, while accessibility trees provide roles, names, values, and hierarchical structure. Optical character recognition can recover text, but it may miss semantic grouping. Vision-language models can describe scenes and identify probable interactive elements [4], [5], but they may hallucinate labels or misunderstand disabled controls. Mobile agents have shown that visual understanding alone can drive action selection in smartphone applications, yet they also reveal brittleness when layouts change or when text is rendered in unusual ways [11]. A robust agent should fuse multiple perceptual channels: pixels for appearance, accessibility metadata for semantics, and platform APIs for state verification. This fusion reduces ambiguity but increases engineering complexity and may create inconsistencies when channels disagree. The system must decide which source is authoritative and how to recover when the environment provides no reliable metadata.

Action grounding is the process of mapping an intended operation to a specific interface element and execution method. In desktop environments, the agent might move a cursor, click a coordinate, type text, or invoke a keyboard shortcut. In web environments, it might manipulate the document object model or send network requests. In mobile environments, it might use touch gestures or accessibility actions. Each method has different reliability, observability, and reversibility. Coordinate-based clicking is general but fragile; API-based execution is precise but limited to instrumented applications. Benchmarks such as Mind2Web and WebArena evaluate web agents in realistic tasks that require navigation, form filling, and information retrieval [13], [14]. OSWorld extends evaluation to open-ended desktop tasks in real operating system environments [15], while AndroidWorld provides a dynamic benchmarking environment for mobile agents [16]. These benchmarks are valuable because they expose the gap between language understanding and grounded execution, but they also tend to measure task completion rather than the quality of failure recovery, user consent, or long-term maintainability.

Tool use is a critical intermediate layer between perception and action. An agent may call search engines, file managers, code interpreters, calendars, or domain-specific APIs instead of simulating every operation through the graphical interface. Tool use improves efficiency and reliability, but it introduces dependency management, versioning, and security challenges. A tool that is safe in one context may be dangerous in another; a file deletion tool might be acceptable in a sandbox but catastrophic on a production machine. The agent must therefore reason not only about what action achieves the goal but also about whether the action is permitted, reversible, and aligned with user expectations. Human-in-the-loop confirmation can mitigate risk, but excessive confirmation destroys the value of automation. The design challenge is to create graduated autonomy, where low-risk actions proceed automatically, medium-risk actions require notification or delayed execution, and high-risk actions require explicit approval. This graduated model depends on accurate risk classification, which is itself a difficult learning and governance problem.

4. Infrastructure and Deployment Considerations

Deploying multimodal computer-use agents requires infrastructure that goes beyond model serving. The system needs secure virtual machines or containers, screen capture pipelines, input injection mechanisms, network isolation, credential management, and telemetry collection. If the agent operates on a user's primary device, privacy risks increase because screen content may include personal messages, financial data, or health information. If it operates in a cloud sandbox, latency and data transfer costs increase, and some applications may not be accessible. Edge deployment can reduce latency and improve privacy, but it limits model size and may require hybrid architectures in which perception runs locally and heavy reasoning runs remotely. The deployment topology therefore shapes the agent's capabilities, cost profile, and risk surface. A system that performs well in a controlled benchmark may fail in an enterprise environment with multi-factor authentication, legacy software, and strict compliance requirements.

State management is another infrastructural challenge. GUI tasks are often stateful and long-running. The agent may need to remember previous screens, unresolved errors, user preferences, and intermediate results. Memory can be stored locally, in a vector database, or in a structured task graph. Each choice affects privacy, auditability, and performance. Local memory improves privacy but complicates synchronization across devices. Cloud memory enables continuity but creates data retention obligations. Audit logs are essential for

debugging and accountability, but they can also become sensitive datasets that require encryption, access control, and retention limits. The infrastructure must support replay, rollback, and forensic analysis. Without these capabilities, organizations cannot determine why an agent acted, whether it was manipulated, or how to prevent recurrence. The socio-technical system therefore depends on observability as much as intelligence.

Cross-domain comparisons reveal different trade-offs. Web agents benefit from structured markup, standardized protocols, and relatively easy sandboxing through headless browsers. Desktop agents face greater heterogeneity, including native applications, modal dialogs, and operating system permissions. Mobile agents operate under strict app sandboxing, touch interaction models, and battery constraints. These differences mean that a single architecture may not be optimal across all environments. A layered design with environment-specific adapters is more practical. The adapters can abstract action spaces, expose state information, and enforce platform-specific policies. This modularity also supports incremental deployment, where an agent begins with limited permissions and expands as trust is established. Recent generalist agent research suggests that unified models can generalize across tasks, but production systems will still require environment-specific engineering to meet reliability, security, and compliance requirements [12]. The infrastructure is not a neutral substrate; it actively shapes what the agent can do and what it can be trusted to do.

5. Robustness, Safety, and Governance

Safety in autonomous computer interaction is not limited to preventing physical harm. It includes protecting data, preserving user intent, avoiding irreversible actions, preventing unauthorized access, and maintaining system integrity. Concrete problems in AI safety, such as reward hacking, negative side effects, and scalable oversight, are directly relevant because GUI agents optimize for task completion and may find unintended shortcuts [17]. For example, an agent asked to delete temporary files might remove important documents if it misclassifies them. An agent asked to fill a form might submit it prematurely. An agent asked to find information might exfiltrate sensitive data through a search query. These failures are not merely model errors; they are governance failures if the system lacks permission boundaries, confirmation thresholds, and rollback mechanisms. Robustness must therefore be engineered at the system level, with redundant checks, policy enforcement, and human escalation paths.

Adversarial risks are particularly acute in GUI environments. Prompt injection can occur when a webpage, document, or email contains instructions that the agent interprets as part of its task. Interface deception can occur when malicious elements mimic trusted controls. Credential theft can occur when the agent enters passwords into the wrong field. Supply chain risks can occur when a tool or plugin is compromised. These threats require defenses that combine model-level robustness, input sanitization, content provenance, and runtime monitoring. The agent should treat external content as untrusted by default, especially when it contains instructions. It should also operate under least privilege, with access limited to the applications and data needed for the current task. Sandboxing can contain damage, but it may also prevent legitimate access. The policy challenge is to define permissible boundaries without making the agent useless. Governance frameworks must specify who is accountable when an agent acts, how incidents are reported, and what remedies exist for affected users.

Fairness and equity are also governance concerns. GUI agents may perform better on popular applications with abundant training data and worse on niche, localized, or assistive technologies. They may misinterpret dialects, non-standard terminology, or culturally specific

interface conventions. The dangers of large-scale language models include bias, environmental cost, and unexamined assumptions about language and community [18]. If computer-use agents are deployed in employment, education, healthcare, or public services, unequal performance can translate into unequal access and outcomes. Governance should therefore require bias evaluation, accessibility testing, and inclusive design. It should also consider the distribution of benefits. If only well-resourced organizations can afford reliable agents, automation may concentrate productivity gains and disadvantage smaller firms, public institutions, and individual users. Fairness is not an add-on to technical development; it is a design constraint that affects data collection, model selection, deployment contexts, and support obligations.

6. Evaluation, Human Factors, and Accountability

Evaluation of multimodal computer-use agents has advanced rapidly through benchmarks such as Mind2Web, WebArena, OSWorld, and AndroidWorld [13], [14], [15], [16]. These benchmarks provide realistic tasks, diverse environments, and reproducible metrics. However, they often emphasize task success, step efficiency, and robustness to layout changes. They may underrepresent long-horizon maintenance, multi-user collaboration, accessibility, privacy leakage, and failure recovery. A high success rate on a benchmark does not guarantee that an agent is safe or fair in a hospital, bank, or government office. Evaluation must therefore include adversarial testing, red-teaming, privacy impact assessment, and human factors research. It should measure not only whether the agent completes tasks but also how users understand its actions, correct its mistakes, and retain meaningful control.

Human factors are central to adoption. Users need mental models of what the agent can do, what it cannot do, and when it is uncertain. Interfaces should communicate intent, progress, and risk without overwhelming the user. Confirmation dialogs must be meaningful, not reflexive. Error messages should explain what went wrong and how to recover. If the agent acts silently in the background, users may lose situational awareness and fail to detect harmful behavior. If it asks for too much confirmation, users may habituate and approve without reading. The design challenge is to calibrate trust so that users rely on the agent appropriately. This requires transparency about capabilities and limitations, as well as clear accountability when failures occur. Organizations deploying agents should establish incident response procedures, training requirements, and review mechanisms that treat the agent as part of a larger workflow rather than an autonomous black box.

Accountability is complicated by the multi-party nature of agent deployment. Model providers, application vendors, infrastructure operators, deployers, and users all contribute to outcomes. A failure may arise from a model hallucination, an application update, a misconfigured permission, or a user's ambiguous instruction. Legal and ethical frameworks must assign responsibility without creating impossible burdens. Contracts, service-level agreements, audit logs, and certification schemes can help clarify roles. Technical standards for logging, permission models, and interoperability can reduce ambiguity. However, standards must be flexible enough to accommodate rapid innovation. The governance challenge is to create accountability structures that are proportionate, transparent, and enforceable. This is especially important when agents operate in regulated domains such as finance, healthcare, and public administration, where errors can have serious consequences.

7. Sustainability, Economics, and Organizational Implications

The sustainability of multimodal computer-use agents depends on computational cost, energy use, and infrastructure lifecycle. Large multimodal models require substantial training and inference resources. If every GUI action triggers a full multimodal inference, operational costs and energy consumption may become prohibitive. System designers must therefore consider caching, model distillation, smaller specialized models, and hybrid architectures that reserve large models for difficult reasoning steps. Cloud data centers have efficiency advantages, but they also require network transmission and cooling. Edge devices can reduce transmission, but they may increase aggregate energy use if inefficient. Sustainability is not only an environmental issue; it is also an economic constraint that determines which organizations can deploy agents at scale. A system that is accurate but too expensive for public institutions and small enterprises may deepen digital divides.

Organizational implications extend beyond cost. Agents alter workflows, job roles, and decision rights. They may automate routine tasks, freeing workers for higher-level judgment, but they may also displace labor or create new forms of algorithmic management. Successful adoption requires process redesign, training, and clear escalation paths. Employees need to understand when to trust the agent, how to override it, and how to report failures. Managers need metrics that capture reliability, safety, and user satisfaction, not just speed. IT departments need change management procedures for model updates, permission changes, and integration with legacy systems. The socio-technical system must be designed for continuous learning, because both interfaces and user needs change over time. An agent that works well today may fail after a software update or a policy change.

From an economic perspective, the value of computer-use agents depends on reliability and scope. Narrow agents that automate a few well-defined tasks may deliver immediate return on investment, while generalist agents promise broader flexibility but higher uncertainty. The trade-off resembles earlier automation waves, where specialized tools often outperformed general-purpose platforms in specific domains. However, multimodal large language agents can reduce integration costs by operating through the user interface rather than requiring custom APIs. This makes them attractive for legacy systems and long-tail applications. The challenge is to manage the resulting technical debt. Screen-scraping and coordinate-based actions are brittle. Organizations that rely on them may accumulate fragile automations that require constant maintenance. A sustainable strategy combines GUI interaction with API integration where possible, and treats agent behavior as code that must be tested, versioned, and governed.

8. Policy, Regulation, and Forward-Looking Perspectives

Policy frameworks for autonomous computer interaction are still emerging. Existing regulations for data protection, cybersecurity, accessibility, and consumer protection provide relevant principles, but they may not address agent-specific risks. Regulators may need to clarify liability for autonomous actions, require transparency about automated decision-making, and establish minimum security standards for agents that access personal data. Certification schemes could evaluate sandboxing, permission controls, audit logging, and incident response. Regulatory sandboxes could allow controlled experimentation while protecting users. International coordination will be important because agents may operate across jurisdictions and cloud regions. However, overly prescriptive rules may freeze architectures before best practices are known. A better approach combines outcome-based requirements with technical standards and ongoing monitoring.

Technical standards can support governance by defining interfaces for permissions, logging, and human override. For example, a standard permission model could specify what an agent is allowed to read, write, click, or transmit. A standard audit format could enable independent review. A standard interruption protocol could allow users or administrators to halt an agent immediately. These standards would not guarantee safety, but they would make accountability more feasible. They would also reduce integration costs and encourage competition on reliability rather than lock-in. Researchers should collaborate with practitioners, policymakers, and civil society to develop standards that are transparent, inclusive, and adaptable. The goal is not to constrain innovation but to ensure that innovation proceeds within boundaries that protect users and institutions.

Forward-looking perspectives must also consider the possibility of increasingly autonomous agents that coordinate with other agents. Multiple agents might negotiate tasks, share context, or delegate subtasks. This could improve efficiency but also create emergent behavior that is difficult to predict or govern. Agent-to-agent interaction may amplify errors, spread misinformation, or create opaque supply chains of automation. Governance frameworks must therefore address not only single-agent safety but also multi-agent system dynamics. Research is needed on identity, authentication, reputation, and conflict resolution among agents. Human oversight must remain meaningful even as systems become more complex. The long-term vision of reliable computer-use agents depends on building infrastructure for trust, not just models for capability. If that infrastructure is neglected, the result may be brittle automation, eroded privacy, and public distrust.

9. Conclusion

Multimodal large language agents for autonomous GUI understanding and computer interaction represent a significant step toward flexible, general-purpose automation. They combine visual perception, language reasoning, memory, planning, and action execution to operate software environments that were designed for humans. This paper has argued that their development must be understood as a system-level challenge involving architecture, infrastructure, governance, sustainability, fairness, and policy. Model accuracy alone is insufficient. Trustworthy agents require layered perception, robust action grounding, sandboxed execution, least-privilege permissions, transparent telemetry, human oversight, and continuous evaluation. They must be assessed not only on task success but also on failure recovery, accessibility, privacy, and accountability. The socio-technical infrastructures that support them will determine whether they augment human capability or introduce new forms of fragility and inequality.

The path forward requires interdisciplinary collaboration. Computer scientists must design modular architectures and rigorous benchmarks. Human-computer interaction researchers must study trust, control, and error recovery. Legal and policy scholars must clarify liability and regulatory expectations. Organizations must invest in governance, training, and incident response. Users must be empowered to understand and override agent behavior. If these efforts are aligned, multimodal computer-use agents could expand access to software, reduce repetitive work, and support people with diverse needs. If they are neglected, the same agents could become vectors for security breaches, opaque automation, and unfair outcomes. The central task is therefore not merely to build agents that can click, type, and navigate. It is to build infrastructures of accountability that make autonomous computer interaction safe, equitable, and sustainable.

References

1. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529-533.
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998-6008.
3. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877-1901.
4. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. *Proceedings of the 38th International Conference on Machine Learning*, 139, 8748-8763.
5. Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., ... & Simonyan, K. (2022). Flamingo: A visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35, 23716-23736.
6. Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., ... & Florence, P. (2023). PaLM-E: An embodied multimodal language model. *Proceedings of the 40th International Conference on Machine Learning*, 202, 8469-8488.
7. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*.
8. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 8634-8652.
9. Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, Article 2, 1-22.
10. Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., ... & Wen, J.-R. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), 186345.
11. Zhang, C., Yang, Z., Liu, J., Han, Y., Chen, X., Huang, Z., ... & Wu, C. (2024). AppAgent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*.
12. Lei, B., Kang, W., Zhang, Z., Chen, W., Xie, X., Zuo, S., ... & Ding, C. (2026). InfantAgent-Next: A Multimodal Generalist Agent for Automated Computer Interaction. *Advances in Neural Information Processing Systems*, 38, 34494-34520.
13. Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., ... & Su, Y. (2023). Mind2Web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36, 28091-28114.
14. Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., ... & Neubig, G. (2023). WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

15. Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., ... & Su, Y. (2024). OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. arXiv preprint arXiv:2404.07972.
16. Rawles, C., Li, A., Rodriguez, D., Riva, O., & Lillicrap, T. (2024). AndroidWorld: A dynamic benchmarking environment for autonomous agents. arXiv preprint arXiv:2405.14573.
17. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565.
18. Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, 610-623.