

# Vision-Language Agents for Adaptive Task Planning in Automated Software Workflows

Gustavo Zaccardo

School of Computing, Clemson University, Clemson, SC, USA.  
gustavozaccardo71@clemson.edu

Curro Valera

Department of Computer Science, George Mason University, Fairfax, VA, USA.  
hellocurro@gmu.edu

Apollonia Zahn

Department of Computer Science, University of Alabama at Birmingham, Birmingham, AL, USA.  
contactapollonia@uab.edu

## Abstract

Vision-language agents combine visual perception, linguistic reasoning, and tool use to plan and execute tasks inside graphical and code-based software environments. This paper examines their role in adaptive task planning for automated software workflows, focusing on system architecture, infrastructure, governance, robustness, fairness, sustainability, and policy. It argues that the central challenge is not merely model capability but the alignment of stochastic multimodal reasoning with deterministic workflow semantics, compliance requirements, and human accountability. The paper develops a layered account of perception, state representation, planning, execution, verification, and observability, showing how each layer introduces trade-offs between flexibility and control. It then analyzes deployment concerns, including latency, cost, energy, dependency management, and technical debt, and considers how model cards, datasheets, audit trails, and human oversight can support responsible automation. The discussion extends to cross-domain comparisons with embodied agents, robotic process automation, and multiagent systems, illustrating where vision-language planning offers advantages and where it creates new failure modes. The conclusion proposes that sustainable adoption depends on treating agents as governed socio-technical actors embedded in institutional processes rather than as standalone software components. The paper contributes a system-level research agenda for designing adaptive, auditable, and equitable vision-language automation in software workflows.

## Keywords

vision-language agents; automated software workflows; adaptive task planning; multimodal interfaces; governance; robust systems; socio-technical infrastructure; human oversight.

## 1. Introduction

Automated software workflows have traditionally relied on deterministic orchestration, scripted integration, and narrow application programming interfaces. These mechanisms remain effective when tasks are stable, inputs are structured, and failure modes can be enumerated in advance. Contemporary software work, however, increasingly spans heterogeneous interfaces, legacy applications, browser-based services, local development

environments, and collaboration platforms that were not designed for machine-readable automation. This heterogeneity creates a demand for agents that can perceive visual state, interpret natural-language goals, reason over partially specified procedures, and adapt plans when the environment changes. Vision-language agents are significant because they promise a bridge between the semantic flexibility of large models and the operational specificity of software systems. Yet that promise also exposes structural tensions between open-ended reasoning and the accountability requirements of production workflows [1], [3], [4].

The research problem is therefore not simply whether a vision-language model can click a button or generate a script. It is whether an agent can participate in a workflow as a dependable actor under conditions of uncertainty, policy constraint, and organizational consequence. Software workflows often carry legal, financial, security, or safety implications, and their automation cannot be evaluated only through task completion rates. A system that executes a correct action in one context may violate separation-of-duties rules in another, leak sensitive information through a screenshot, or create an unauditable chain of decisions. Adaptive planning introduces further complexity because the agent may modify its own trajectory, invoke external tools, or reinterpret ambiguous instructions in ways that were not anticipated by workflow designers. Understanding these systems requires an interdisciplinary lens that treats them as socio-technical infrastructures rather than as isolated model deployments [13], [18].

This paper develops a system-level account of vision-language agents for adaptive task planning in automated software workflows. It begins with conceptual foundations drawn from situated action, multiagent systems, and multimodal learning. It then examines architectural layers, planning mechanisms, governance structures, deployment constraints, robustness and fairness concerns, evaluation practices, and cross-domain comparisons. The central argument is that sustainable automation depends on designing agents as governed participants within an operational envelope, not as autonomous replacements for human judgment. Their value emerges when probabilistic multimodal reasoning is constrained by verifiable interfaces, observability, policy enforcement, and clear escalation paths. The paper concludes with a forward-looking agenda for research and practice that balances adaptability with institutional control.

## **2. Conceptual Foundations and Related Work**

The conceptual roots of adaptive task planning reach back to situated action theory, which challenged the idea that plans fully determine behavior. In complex environments, actors continuously interpret local conditions and adjust routines in response to contingencies that formal plans cannot completely anticipate [1]. This insight is directly relevant to software workflows, where graphical interfaces, network latency, permission prompts, and unexpected dialogs create situated conditions that scripts often fail to handle. Multiagent systems research adds another foundation by framing computation as coordinated action among autonomous or semi-autonomous entities with goals, capabilities, and communication protocols [2]. Vision-language agents inherit this tradition because they must negotiate tool access, share state, and sometimes collaborate with other agents or human operators. Classical artificial intelligence contributes planning formalisms, search, and knowledge representation, while modern deep learning supplies perceptual and linguistic generalization [3], [4].

The technical foundation of current vision-language agents rests on representational learning and attention-based architectures. Deep neural networks enable hierarchical feature extraction across images, text, and structured data, while transformer models allow long-range

dependencies and flexible multimodal fusion [4], [5]. Vision-language pretraining further aligns images and text in a shared embedding space, supporting zero-shot recognition and cross-modal retrieval [6]. Large language models contribute few-shot reasoning, instruction following, and in-context adaptation [7]. When these capabilities are combined with action selection, the resulting agents can interleave reasoning traces with tool calls and environmental feedback, a pattern that improves transparency and adaptability in interactive tasks [8]. More recent multimodal generalist agents extend this pattern to automated computer interaction, using visual observations and language instructions to operate graphical interfaces across applications [9]. Embodied multimodal models demonstrate related capacities in robotics, where perception, language, and action are integrated for physical tasks [10]. Toolformer-style approaches show that language models can learn when and how to invoke external tools, which is essential for software workflows that depend on APIs, databases, and command-line utilities [11].

However, these foundations also reveal unresolved tensions. Language models are shaped by statistical regularities in training data, not by explicit workflow semantics, so their plans may be plausible but operationally invalid [17]. Instruction tuning and human feedback can improve alignment with user intent, but they do not guarantee compliance with domain-specific policy [12]. Foundation models concentrate capability and risk, raising questions about transparency, dependency, and governance [18]. Safety research has long identified specification gaming, reward hacking, side effects, and distributional shift as core challenges for autonomous systems [19]. In software workflows, these challenges manifest as incorrect tool invocation, unauthorized data access, brittle recovery behavior, and silent divergence from organizational procedure. Consequently, vision-language agents should be understood as components within a larger control problem, not as standalone reasoning engines. Their planning behavior must be coupled to verification, policy, and human escalation mechanisms if they are to be trusted in consequential environments.

### **3. Architectural Layers and System Boundaries**

A useful architecture for vision-language workflow agents separates perception, state representation, planning, execution, verification, and observability. Perception includes screenshots, accessibility trees, document content, logs, and natural-language instructions. State representation transforms these heterogeneous signals into a working model of the current task, the available tools, user permissions, and environmental constraints. Planning selects goals and subtasks, while execution maps decisions to concrete actions such as clicks, keystrokes, API calls, or code edits. Verification checks whether the action achieved the intended effect and whether it respected policy. Observability records traces, evidence, and metrics for debugging and audit. Each layer introduces trade-offs. Richer perception improves grounding but increases latency, cost, and privacy exposure. More detailed state representation supports better reasoning but creates synchronization challenges when the environment changes. Stronger planning flexibility enables adaptation but complicates verification and reproducibility.

System boundaries are equally important. An agent may operate within a single application, across a suite of enterprise tools, or at the level of an operating system. Narrow boundaries simplify permission management and testing, but they limit the agent's ability to complete end-to-end workflows. Broad boundaries increase utility but expand the attack surface and make it harder to assign responsibility when failures occur. The choice of boundary also determines whether the agent can observe sensitive information such as personal data,

financial records, or security credentials. In practice, architectures tend to adopt mediated access through tool gateways, sandboxed execution environments, and role-based permissions. These mediators can enforce policy, normalize interfaces, and provide a stable abstraction layer despite changes in underlying applications. Yet every mediation layer adds engineering complexity and potential bottlenecks. The architectural question is therefore not whether to mediate, but where to place mediation so that safety, adaptability, and operational efficiency remain balanced [13], [14].

Deployment architecture also shapes failure modes. Cloud-hosted agents can leverage powerful multimodal models and centralized logging, but they introduce network dependency and data residency concerns. Local or edge-hosted agents reduce data exposure and latency but may lack sufficient compute for high-fidelity perception. Hybrid designs route sensitive perception locally and complex reasoning remotely, but they create synchronization and policy consistency challenges. Multiagent designs distribute planning across specialized roles, which can improve modularity and resilience, but they also produce emergent coordination failures that are difficult to diagnose [2]. A robust system must therefore specify not only model components but also fallback behavior, retry limits, human handoff conditions, and shutdown procedures. These are governance decisions as much as engineering decisions because they determine who or what retains control when uncertainty exceeds acceptable thresholds.

#### **4. Adaptive Task Planning and Multimodal Grounding**

Adaptive task planning in software workflows requires the agent to convert an underspecified goal into an executable sequence while remaining responsive to environmental feedback. A user might ask an agent to reconcile a spreadsheet, update a customer record, and notify a team, but the specific applications, permissions, and formatting rules may vary. The agent must ground language in visual and structural evidence, such as identifying the correct spreadsheet column, recognizing a disabled button, or inferring that a modal dialog requires confirmation. Multimodal grounding is therefore not a peripheral capability; it is the mechanism by which abstract intent becomes situated action [6], [10]. The agent must also decide when to ask for clarification, when to proceed with a safe default, and when to abort because the risk of misinterpretation is too high. These decisions require an explicit notion of confidence, risk, and reversibility that many current models do not natively provide.

Tool use is central to adaptive planning because software workflows rarely consist of purely visual interactions. Agents may need to invoke APIs, run scripts, query databases, or modify code. Toolformer-style learning demonstrates that models can acquire tool invocation behavior, but production workflows require stronger guarantees about argument validity, authentication, rate limits, and side effects [11]. A vision-language agent might combine graphical interaction with programmatic calls, using the interface when no API exists and using the API when precision and speed are required. This hybrid approach improves capability but complicates traceability because actions span different modalities and permission domains. Verification must therefore correlate visual observations with API responses and system logs. Without such correlation, an agent may believe it succeeded while the underlying system remained unchanged. The planning loop must include not only action selection but also evidence collection and reconciliation.

Adaptive planning also raises questions about autonomy levels. At low autonomy, the agent proposes actions and waits for approval. At medium autonomy, it executes reversible steps independently and requests approval for consequential actions. At high autonomy, it executes

end-to-end workflows with post hoc review. Each level shifts the burden of vigilance between human and machine. Higher autonomy can reduce labor costs and latency, but it also increases the chance that errors propagate before detection. In regulated domains, the acceptable autonomy level may be constrained by law or policy rather than by technical capability. A well-designed system should make autonomy configurable at the workflow, role, and action level. It should also support progressive trust, where successful performance on low-risk tasks earns expanded permissions under continuous monitoring [14], [15]. Adaptive planning is thus inseparable from adaptive governance.

## **5. Governance, Policy, and Accountability**

Governance for vision-language agents must address both model behavior and workflow outcomes. Model behavior includes accuracy, bias, calibration, and robustness, while workflow outcomes include compliance, security, service quality, and equitable access. Traditional software governance relies on version control, testing, and change management, but machine learning systems introduce additional sources of technical debt, including data dependencies, feedback loops, configuration entanglement, and hidden assumptions [13]. Vision-language agents intensify these issues because their behavior depends on prompts, model versions, tool interfaces, and environmental state. A workflow that passes testing today may fail after an application update, a model refresh, or a subtle change in user language. Governance must therefore include continuous monitoring, drift detection, and rollback procedures. It must also define who is accountable when an agent acts incorrectly: the model provider, the system integrator, the workflow owner, or the human operator who approved the action. Clear allocation of responsibility is essential for both legal compliance and organizational trust.

Policy enforcement should be embedded in the agent architecture rather than bolted on after deployment. Access control, data minimization, purpose limitation, and audit logging can be implemented at the tool gateway, execution sandbox, or planning layer. For example, an agent may be permitted to read customer records but not export them, or to modify draft documents but not publish final versions. These constraints can be represented as policies that the planner must satisfy before executing an action. When conflicts arise, the agent should escalate rather than improvise. Model cards and datasheets provide structured documentation for models and datasets, supporting transparency about intended use, limitations, and evaluation conditions [15], [16]. However, documentation alone does not enforce policy. It must be coupled with runtime controls, periodic audits, and meaningful human oversight. Otherwise, governance becomes a compliance artifact rather than an operational safeguard.

Accountability also requires contestability. Users affected by automated decisions should have mechanisms to understand, challenge, and correct outcomes. In software workflows, this may mean preserving an audit trail that links each action to a model version, input state, policy check, and human approval. It may also mean providing explanations that are meaningful to non-specialists without exposing sensitive system details. Explanation is difficult for multimodal agents because their reasoning is distributed across perception, language, and action. A useful approach is to surface evidence rather than internal model states: the screenshot region used, the field value read, the rule applied, and the tool response received. This evidence-based accountability aligns with established fairness and transparency practices while remaining practical for engineering teams [15], [17]. Governance should not aim to eliminate all uncertainty, but to ensure that uncertainty is visible, bounded, and subject to appropriate human judgment.

## 6. Deployment, Infrastructure, and Sustainability

Deploying vision-language agents at scale requires infrastructure that can handle variable latency, high computational cost, and dynamic tool environments. Multimodal inference is often more expensive than text-only inference because it processes images, layout information, and long context windows. Workflows that require frequent screenshots or real-time interaction may face latency constraints that degrade user experience or cause timeouts. Infrastructure design must therefore consider caching, batching, model distillation, and selective perception. Not every step requires full multimodal reasoning; some can be handled by deterministic scripts or lightweight classifiers. A tiered architecture can route simple actions to fast components and reserve large models for ambiguous or high-risk decisions. This improves sustainability by reducing energy consumption and cost while preserving adaptability where it matters most [13], [18].

Sustainability also includes maintainability and organizational capacity. Vision-language agents depend on prompts, tool definitions, model endpoints, and evaluation datasets that must be versioned and updated. If these dependencies are poorly managed, the system accumulates hidden technical debt and becomes fragile. Teams need monitoring dashboards, regression tests, incident response playbooks, and clear ownership. The ML test score rubric offers a useful starting point for production readiness by emphasizing data integrity, model reliability, and monitoring [14]. However, agentic systems require additional checks for action safety, tool authorization, environmental drift, and recovery behavior. Deployment should be staged, beginning with shadow mode where the agent observes human actions without executing, then moving to assisted mode with approval gates, and only later to limited autonomy. This staged approach allows organizations to build evidence and trust while containing the blast radius of failures.

The environmental and economic sustainability of large-scale agent deployment is another concern. Foundation models consume substantial energy during training and inference, and widespread automation could increase aggregate demand if efficiency gains lead to more intensive use [17], [18]. Organizations should therefore measure cost and energy per completed workflow, not only per model call. They should also consider whether an agent is genuinely necessary for a task or whether simpler automation would suffice. In many cases, the most sustainable design combines deterministic automation for stable steps with vision-language reasoning for exceptions. This hybrid strategy reduces waste and improves reliability. Infrastructure choices are not neutral; they shape access, equity, and the distribution of benefits. If only well-resourced organizations can deploy capable agents, automation may deepen existing asymmetries in productivity and service quality.

## 7. Robustness, Fairness, and Socio-Technical Risk

Robustness for vision-language agents must address perceptual noise, interface changes, ambiguous instructions, and adversarial manipulation. A minor visual change, such as a relocated button or altered label, can cause catastrophic failure if the agent has overfit to a previous layout. Adversarial inputs may include misleading text in documents, hidden interface elements, or prompt injection through external content. Because agents often combine trusted instructions with untrusted observations, they are vulnerable to confusion between data and commands. Safety research has identified related problems in autonomous systems, including negative side effects, reward hacking, and unsafe exploration [19]. In software workflows, these problems can manifest as unauthorized data disclosure, erroneous financial transactions, or destructive code changes. Robustness therefore requires adversarial

testing, input sanitization, privilege separation, and conservative handling of untrusted content. It also requires graceful degradation, so that the agent stops or seeks help rather than continuing with low confidence.

Fairness concerns arise at multiple levels. At the model level, vision-language systems may perform unevenly across languages, interfaces, document layouts, and cultural conventions. At the workflow level, automation may disproportionately affect workers whose tasks are most easily observed and standardized, while shifting remaining work to monitoring and exception handling. At the societal level, agentic automation may influence access to services, employment conditions, and the distribution of decision-making power. These are not merely technical biases; they are structural consequences of deployment choices [17], [18]. Fairness-aware design should include representative evaluation data, participatory impact assessment, and ongoing monitoring for disparate outcomes. It should also consider who benefits from increased efficiency and who bears the cost of errors. In software workflows, fairness may require ensuring that automated processes do not disadvantage users with disabilities, limited connectivity, or non-dominant languages.

Socio-technical risk is amplified by opacity and automation bias. Human operators may over-trust agent recommendations, especially when interfaces present actions as authoritative. Conversely, excessive alerting can cause operators to ignore warnings. Designing effective human-agent collaboration requires careful attention to workload, expertise, and incentive structures. The agent should communicate uncertainty, rationale, and consequences in ways that support informed intervention. It should also adapt its autonomy to the operator's role and the criticality of the task. In high-stakes environments, human oversight must be meaningful rather than ceremonial. This means giving operators enough time, information, and authority to override the agent. It also means training them to recognize common failure modes and to document interventions. Robustness and fairness are therefore not separate from governance; they are outcomes of governance working well in practice.

## **8. Evaluation and Observability**

Evaluating vision-language agents for software workflows requires metrics that go beyond task success. Useful dimensions include completion accuracy, step efficiency, policy compliance, recovery from failure, human intervention rate, latency, cost, and user trust. Evaluation should occur in simulation, shadow mode, and controlled production environments because laboratory benchmarks rarely capture the complexity of real interfaces and permissions. Observability infrastructure must capture multimodal traces, tool calls, state changes, policy decisions, and human approvals. These traces support debugging, audit, and continuous improvement. They also enable the construction of regression tests from real incidents. Without observability, organizations cannot know whether an agent is improving or silently accumulating risk. Observability is thus a precondition for both safety and learning [13], [14].

Evaluation must also account for distributional shift. Software interfaces change, policies evolve, and user language drifts. An agent that performed well last quarter may fail after an application update or a new compliance rule. Continuous evaluation should monitor for shifts in input distribution, action distribution, and outcome quality. It should include adversarial scenarios, red-team exercises, and edge cases involving ambiguous or conflicting instructions. Benchmarks should be transparent about their assumptions and limitations, and they should include measures of equity across user groups. Model cards and datasheets can document evaluation conditions, but runtime telemetry is needed to detect when those conditions no

longer hold [15], [16]. A mature evaluation practice treats every production incident as data for improving both the model and the surrounding workflow.

## **9. Case Illustrations and Cross-Domain Comparisons**

Consider an enterprise accounts payable workflow in which an agent must review invoices, match them to purchase orders, enter data into a financial system, and route exceptions to human specialists. Vision-language capabilities help the agent interpret scanned invoices, navigate legacy interfaces, and handle variations in vendor formats. Adaptive planning allows it to adjust when a purchase order is missing or a field is ambiguous. However, financial controls require segregation of duties, audit trails, and approval thresholds. The agent must therefore operate within a policy envelope that limits payment authorization and preserves evidence. A purely model-centric design would fail these requirements; a governed architecture that combines perception, policy checks, and human escalation is more likely to succeed. This case illustrates that the value of vision-language agents lies in handling variability within a controlled process, not in removing control altogether.

A second illustration involves software development workflows, where agents assist with issue triage, code edits, test execution, and deployment preparation. Open-ended embodied agents have demonstrated the ability to explore environments, acquire skills, and pursue long-horizon goals with language guidance [20]. Similar principles can apply to development environments, where an agent reads issues, inspects repositories, runs tests, and proposes patches. Yet software workflows have strong version control, code review, and continuous integration practices that create natural checkpoints. These checkpoints can be used as verification gates for agentic actions. The agent may prepare a branch and open a pull request, but humans or automated policy engines approve merges. This hybrid model preserves the adaptability of vision-language planning while leveraging existing software governance. It also highlights a cross-domain lesson: domains with strong process instrumentation are better positioned to absorb agentic automation safely than domains with opaque or informal workflows.

Cross-domain comparisons with robotic process automation and embodied robotics clarify the distinctive challenges of software agents. Robotic process automation excels at deterministic, high-volume tasks but struggles with interface variation and unstructured inputs. Embodied agents face physical uncertainty, safety constraints, and real-time control requirements. Vision-language software agents occupy a middle ground: they operate in symbolic-digital environments that are more controllable than physical worlds but more heterogeneous than traditional APIs. This middle ground offers opportunities for fast iteration, sandboxing, and detailed logging. It also creates risks of overconfidence because digital actions can propagate instantly across systems. The most promising designs borrow from all three domains: the reliability engineering of robotic process automation, the multimodal grounding of embodied agents, and the policy and audit frameworks of enterprise software. Such synthesis moves the field beyond model benchmarking toward systems that are deployable, governable, and sustainable.

## **10. Conclusion**

Vision-language agents for adaptive task planning in automated software workflows represent a significant step toward flexible automation, but their value depends on architectural discipline and institutional governance. The ability to perceive interfaces, interpret language, use tools, and adjust plans can address longstanding limitations of scripted automation. Yet

the same capabilities introduce new failure modes, including policy violations, opaque decision chains, unfair distribution of impacts, and unsustainable infrastructure demands. This paper has argued that these challenges cannot be solved by model improvement alone. They require layered architectures, runtime policy enforcement, observability, human oversight, and clear accountability. They also require a shift in perspective: agents should be treated as governed participants in socio-technical systems rather than as autonomous replacements for human judgment.

The research agenda ahead should prioritize verifiable planning, uncertainty-aware action selection, policy-aware tool use, and evaluation methods that capture real workflow consequences. It should also examine the organizational conditions under which automation improves work rather than merely displacing it. Cross-domain learning from multiagent systems, robotics, and production machine learning can inform these efforts. Ultimately, the most successful vision-language agents will be those that combine adaptability with restraint, expanding what can be automated while preserving the control, fairness, and accountability that software workflows require.

## References

1. Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communication*. Cambridge University Press.
2. Wooldridge, M. (2009). *An introduction to multiagent systems* (2nd ed.). Wiley.
3. Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.
4. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.
5. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998-6008.
6. 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.
7. 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.
8. 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*.
9. 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.
10. 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, 8469-8484.
11. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language models can teach

themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 68539-68551.

12. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., ... & Lowe, R. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35, 27730-27744.
13. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, 28, 2503-2511.
14. Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017). The ML test score: A rubric for ML production readiness and technical debt reduction. *2017 IEEE International Conference on Big Data*, 1123-1132.
15. Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., ... & Gebru, T. (2019). Model cards for model reporting. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 220-229.
16. Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé III, H., & Crawford, K. (2021). Datasheets for datasets. *Communications of the ACM*, 64(12), 86-92.
17. 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.
18. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... & Liang, P. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.
19. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.
20. Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, T., Zhu, Y., Fan, L., & Anandkumar, A. (2023). Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.