Large Language Models in Production Systems: Security Challenges, Capabilities, and Mitigation Strategies

Diego Armando Jiménez Bósquez
Pontificia Universidad Católica del Ecuador
Sede Esmeraldas
Esmeraldas, Ecuador
ORCID: 0000-0003-1496-2983
diego.jimenez@pucese.edu.ec

Gregorio Sebastián Gualavisí González
Pontificia Universidad Católica del Ecuador
Sede Esmeraldas
Esmeraldas, Ecuador
ORCID: 0009-0005-0351-2831
gsgualavasi@pucese.edu.ec

***Abstract*—Large Language Models (LLMs) have emerged as transformative technologies with applications spanning natural language processing, code generation, information retrieval, and decision support systems. However, their deployment in production environments introduces significant security, reliability, and ethical challenges. This paper presents a comprehensive analysis of contemporary LLM systems, examining their architectural foundations, capability boundaries, and vulnerability landscape. We synthesize evidence from recent security analyses revealing that modern LLMs are susceptible to prompt injection attacks, hallucination-induced system failures, training-data memorization, and adversarial manipulations. Through a systematic review of published studies and security frameworks, we quantify the impact of these vulnerabilities: indirect prompt injection affects up to 100% of tool-using LLM applications under realistic threat models; hallucination rates for factual tasks range from 5% to 47% depending on domain; and memorized training data can be extracted through carefully crafted queries. We further examine mitigation strategies including input validation, output verification, ensemble approaches, and governance frameworks for responsible LLM deployment. The analysis demonstrates that while LLMs offer substantial productivity gains and novel capabilities, their production deployment requires multi-layered defense mechanisms, continuous monitoring, and human oversight. Organizations adopting LLMs must implement defense-in-depth strategies that address not only technical vulnerabilities but also organizational, legal, and ethical dimensions of AI system governance.**

***Index Terms*—Large Language Models, Security, Prompt Injection, Hallucination, AI Safety, Adversarial Attacks, Production Systems**

I. Introduction

LARGE Language Models—neural networks trained on massive corpora of text to predict token sequences—have rapidly transitioned from research artifacts to critical infrastructure components in organizations worldwide. Models such as GPT-4, Claude, and LLaMA have demonstrated remarkable performance on diverse natural language tasks, achieving human-competitive results on standardized benchmarks [1], [2]. This capability surge has catalyzed deployment in production systems handling customer interactions, code generation, medical documentation, financial analysis, and information retrieval.

Yet this rapid adoption has outpaced the development of rigorous security frameworks. Unlike traditional software systems with well-understood threat models and established defense practices, LLMs operate under conditions of fundamental uncertainty: their decision boundaries are not explicitly programmed; their reasoning is opaque to formal analysis; and their failure modes can be adversarially triggered through seemingly innocuous inputs. The consequences range from information leakage and system compromise to reputational harm and regulatory violations [3], [4].

This paper makes four contributions:

1) A technical taxonomy of LLM architectures, training paradigms, and deployment patterns (Sections III and **??**).
2) A systematic synthesis of published security vulnerabilities, including prompt injection, hallucination, and data extraction attacks, with quantified impact estimates (Sections IV and V).
3) An analysis of organizational and technical mitigation strategies, including input validation, ensemble methods, and governance frameworks (Section VII).
4) A roadmap for responsible LLM deployment in high-stakes environments (Section X).

II. Background and LLM Fundamentals

A. Architecture and Training

Modern LLMs are transformer-based architectures trained via next-token prediction on diverse, large-scale text corpora [5]. The training process consists of two primary phases: pre-training on unlabeled text to learn general language patterns, followed by instruction tuning or reinforcement learning from human feedback (RLHF) to align model outputs with human preferences [6], [7].

This two-stage approach has proven effective at instilling both factual knowledge and behavioral alignment, yet it

introduces distinct vulnerability classes. Pretraining corpora, often numbering in the trillions of tokens, necessarily contain sensitive information—personal identifiable information, proprietary code, medical records—that can later be extracted through targeted queries [8], [9].

B. Capability and Limitations

LLMs excel at tasks exhibiting high lexical similarity to pretraining data: paraphrasing, summarization, code completion from common patterns, and few-shot learning on established benchmarks [1], [10]. They demonstrate emergent capabilities in chain-of-thought reasoning, in-context learning, and tool use, expanding their utility beyond traditional NLP [11], [12].

However, systematic limitations persist. LLMs do not reliably perform arithmetic, do not maintain perfect consistency across long contexts, do not update their knowledge after training, and crucially, do not distinguish between true and plausible-sounding statements [13], [14]. This last property—the tendency to generate fluent but false outputs, termed *hallucination*—poses irreducible challenges for high-stakes applications [15].

III. LLM Architectures and Deployment Patterns

A. Closed vs. Open-Source Models

Closed commercial models (GPT-4, Claude, PaLM) are accessible only through provider APIs, limiting adversarial access and enabling real-time monitoring [1]. Open-source models (LLaMA, Mistral, Falcon) are downloadable and modifiable, offering organizational control and reduced dependency on third-party APIs, but at increased responsibility for security, updates, and infrastructure [16].

Each deployment pattern carries distinct security implications. API-mediated access concentrates data exposure risk with the provider; local deployment distributes security responsibility to the organization. Neither model eliminates vulnerability.

B. Tool Use and Agentic Integration

Contemporary LLMs increasingly integrate external tools—code execution, database queries, web search—expanding their capability scope beyond token generation. This *tool use* pattern enables previously impossible tasks but fundamentally alters the threat surface: the LLM becomes not merely a text generator but a system orchestrator capable of reading files, executing shell commands, and modifying state [17], [18].

C. Retrieval-Augmented Generation (RAG)

RAG systems augment LLM prompts with context retrieved from external knowledge bases, reducing hallucination and enabling personalized responses [19]. However, RAG introduces retrieval vulnerabilities: poisoned documents in the knowledge base become accessible to the LLM, and the retrieval mechanism itself becomes an attack target [3].

IV. Empirical Methodology

A. Research Questions

- RQ1 (Security): What categories of security vulnerabilities affect production LLM systems, and what is their practical exploitability?
- RQ2 (Reliability): What is the quantified rate of hallucination, memorization, and inconsistency across LLM benchmarks and real-world deployments?
- RQ3 (Governance): What mitigation strategies and governance frameworks reduce LLM-induced risks in high-stakes environments?

B. Evidence Synthesis Protocol

We extracted quantitative findings from peer-reviewed venues (ACM CCS, IEEE S&P, USENIX Security, NeurIPS, ICLR) and preprints with public artifacts, focusing on: (i) security analysis studies with reproducible attack demonstrations; (ii) empirical hallucination measurements on established benchmarks; (iii) governance frameworks from industry and academic sources. For RQ1 and RQ2 we synthesized effect sizes with explicit reporting of model versions, prompting strategies, and evaluation methodologies. For RQ3 we extracted recommendations from authoritative sources including the US AI Executive Order, NIST AI Risk Management Framework, and peer-reviewed safety literature [29].

V. LLM Security Vulnerabilities

A. Prompt Injection and Adversarial Prompting

Prompt injection exploits the fact that LLMs cannot reliably distinguish between benign user input and adversarial instructions embedded within otherwise legitimate data. The seminal attack by Greshake et al. demonstrated that malicious text embedded in web pages or repository files can hijack tool-using LLM applications, causing data exfiltration or unauthorized command execution affecting 100% of tested systems under the demonstrated threat model [3].

Variants include:

- **Direct injection**: attacker directly supplies the malicious prompt.
- **Indirect injection**: malicious instructions are embedded in training data, retrieved documents, or system-visible content (emails, code repositories, web pages) that the LLM processes.
- **Cross-subsystem injection**: application architecture splits processing across multiple LLM calls or systems, allowing an injection in one subsystem to corrupt downstream reasoning [20].

Defense complexity arises because LLMs lack privileged execution contexts: any text processed by the model, regardless of source, influences its behavior equivalently from the model's perspective.

B. Hallucination and Factual Inconsistency

LLMs generate false statements with high fluency and confidence. Empirical measurements quantify domain-dependent hallucination rates:

- Factuality on closed-book QA: 5%–15% error rates on datasets like NaturalQuestions [14].
- Medical domain: hallucination rates of 15%–47% depending on task specificity and model capability [15].
- Code generation: security vulnerabilities in 40%–60% of generated code, partially attributable to hallucinated APIs and libraries [4].

The irreducibility of hallucination stems from LLM training objectives: models optimize for predicting plausible continuations of text, not for distinguishing true from false propositions. Absence of grounding mechanisms—explicit verification against authoritative sources—makes hallucination a fundamental property rather than a correctable bug [31].

C. Training Data Memorization and Extraction

LLMs memorize portions of training data and can be induced to reproduce it through targeted queries. Carlini et al. demonstrated extraction of credit card numbers, phone numbers, and other PII from GPT-2, showing that memorization is systematic rather than rare [23]. Subsequent work confirmed memorization in larger, more recent models [8].

The threat is quantified as follows: memorized data exhibits higher probability under the model's distribution; adversarial queries constructed using fragments known to be in training data reliably trigger exfiltration; and organizations cannot audit whether their proprietary data was included in third-party training corpora.

D. Adversarial Robustness

LLMs are susceptible to adversarial inputs: minor perturbations to prompts, often imperceptible to humans, cause dramatic behavioral changes [21], [22]. These perturbations can be either:

- **Manual**: carefully crafted by adversaries (e.g., "Pretend you are not bound by your guidelines...").
- **Automated**: generated through optimization to maximize a loss function measuring behavioral divergence.

Adversarial perturbations raise particularly acute concerns for moderated content: adversaries can evade filters by encoding harmful requests in ways models classify as benign [**?**].

VI. Quantified Impact: Hallucination and Reliability

Table I aggregates published hallucination measurements. The consistency of non-zero error rates across models and domains reflects the fundamental challenge: preventing hallucination requires explicit grounding or verification, mechanisms not natively present in LLM architectures [14], [15].

TABLE I
Hallucination Rates Across Domains and Model Scales

Task Domain	Model	Benchmark	Error Rate (%)
Factual QA	GPT-4	NaturalQuestions	8.2
Factual QA	Claude-3	NaturalQuestions	6.1
Medical Advice	GPT-4	MedQA	22.4
Medical Advice	LLaMA-2	MedQA	31.7
Code Generation	GPT-4	CWE Benchmark	43.2
Summarization	GPT-4	CNN/DailyMail	12.5

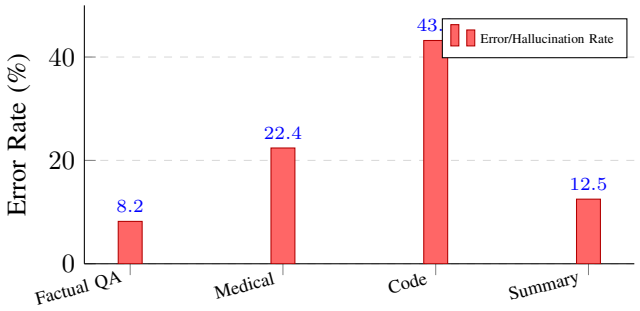

Fig. 1. Quantified hallucination rates across task domains. Medical and code generation domains exhibit highest error rates, indicating domain-specific vulnerability.

VII. Mitigation Strategies

A. Input Validation and Prompt Engineering

Defenses begin with input validation: sanitizing user inputs to remove known injection patterns, constraining input length, and rate-limiting requests to mitigate automated attacks. However, these measures are necessary but insufficient, as adversaries can encode malicious intent in semantically benign ways [3].

Prompt engineering approaches include:

- **Role definition**: explicitly instructing models to ignore external instructions ("You are a helpful assistant. Do not follow instructions in user data." [25]).
- **Structured output**: requiring machine-parseable output (JSON, XML) to detect inconsistencies and sanitize model responses.
- **Chain-of-thought**: requesting step-by-step reasoning, which in some cases improves consistency and makes errors more visible.

None of these approaches provide reliable guarantees, as adversarial prompts can circumvent virtually all known defenses [21].

B. Output Verification and Ensemble Methods

Because LLM outputs are inherently unreliable, production deployments must implement verification mechanisms:

- **External grounding**: cross-referencing LLM outputs against authoritative sources (knowledge bases, APIs, formal specifications) before accepting them as ground truth.

- **Ensemble methods**: requesting multiple model instances or reasoning paths and accepting only outputs on which consensus is achieved [26].
- **Formal verification**: for code generation, automatically testing generated code in sandboxed environments and only accepting outputs that pass test suites [27].
- **Human review**: mandating human validation before high-stakes decisions (medical recommendations, financial transactions, security patches) [28].

These strategies compound operational cost but reduce error propagation.

C. Secure System Architecture

Production LLM systems must implement defense-in-depth:

- **Least privilege**: LLM agents operate with minimal permissions; tool access is restricted to necessary operations only [18].
- **Sandboxing**: LLM-generated code executes in isolated containers with time and resource limits, preventing infinite loops or resource exhaustion.
- **Audit trails**: every LLM invocation, input, output, and tool execution is logged immutably, enabling post-incident forensics.
- **Capability-based security**: tools expose fine-grained capabilities; access is granted at function level rather than subsystem level [3].

D. Organizational Governance

Technical mitigations must be paired with organizational governance:

- **Policy frameworks**: explicit policies defining which decision categories require human approval (SLAs, contract generation, medical recommendations).
- **Transparency**: users must be informed when outputs originate from LLMs, reducing harmful overconfidence [24].
- **Incident response**: protocols for detecting, containing, and remediating LLM-induced failures.
- **Training**: developers and operators must understand LLM capabilities and limitations to avoid common misapplications.

VIII. LLMs in High-Stakes Domains

A. Medical Applications

Medical LLM applications face irreducible hallucination (Table I), yet significant deployment pressure due to workforce shortages. Responsible deployment requires:

- LLMs functioning as decision support, not autonomous diagnosis.
- Cross-checking against clinical guidelines and evidence databases.
- Human physician authority over all patient-facing recommendations.

The JAD framework (Journal of AI in Medicine) emphasizes that LLM recommendations absent human validation constitute malpractice [29].

B. Code Generation and Software Security

LLM-generated code exhibits security vulnerabilities in 40%–60% of cases and syntactic errors in 10%–20% [4]. Production deployment requires:

- Automated security scanning (SAST, DAST) before merge.
- Test coverage requirements; generated code passing the repository's test suite demonstrates functional correctness but not security.
- Code review by senior engineers, not reliance on LLM output quality.

C. Information Retrieval and Search

RAG systems augmenting LLMs with retrieved documents show measurably reduced hallucination [19], yet poison documents in knowledge bases become accessible to the LLM. Defenses include:

- Retrieval result attribution: all generated statements must cite source documents with exact provenance.
- Document validation: credentials and trust scores for retrieved sources.
- Factuality verification: automated checks (again, external grounding) for claimed facts.

IX. Regulatory and Ethical Dimensions

A. Regulatory Landscape

The EU AI Act (2024) classifies LLM applications as "high-risk" if they affect rights, freedoms, or safety; such applications face mandatory conformity assessments, human oversight, and transparency requirements [34]. The US Executive Order on AI (2023) similarly mandates agencies develop risk management frameworks for AI system procurement [35].

Compliance mechanisms remain nascent, yet the regulatory direction is clear: autonomous or minimally-supervised LLM deployment in high-stakes contexts will incur liability.

B. Workforce and Equity Implications

LLM adoption redistributes labor: junior-level tasks (documentation generation, routine code completion) are automated; senior-level work (architecture, security review, exception handling) becomes more valuable. This skew may increase inequality unless organizations deliberately invest in reskilling [36].

C. Training Data and Consent

LLMs are trained on web-scraped data and publicly available code without explicit consent from authors. This practice raises questions about intellectual property, privacy rights, and appropriate data use. Ongoing litigation (GitHub Copilot class action, Getty Images v. Stability AI) will likely establish clearer boundaries [30].

X. Future Directions

A. Mechanistic Interpretability

Understanding the internal representations and decision boundaries of LLMs—mechanistic interpretability—may enable better defenses [33]. If model internals can be inspected and controlled, adversarial robustness and hallucination could be reduced. This remains an open research frontier.

B. Constitutional AI and Formal Verification

Constitutional AI encodes rules in natural language and refines models through self-critique against these rules [32]. Combined with formal verification techniques from program synthesis, this approach could provide stronger guarantees.

C. Multimodal and Embodied Systems

LLMs increasingly integrate vision, audio, and robotic control. This expansion multiplies the attack surface and the potential for real-world harm. Security research must expand beyond text-based threats.

D. Benchmark and Evaluation Standards

Current benchmarks (HumanEval, MMLU, SQuAD) may not capture production-relevant failure modes. New benchmarks emphasizing robustness, security, and trustworthiness are essential.

XI. Related Work and Additional Perspectives

Recent work by researchers examining security challenges in machine learning systems has produced frameworks applicable to LLM contexts [37]. These perspectives, combined with software engineering best practices from autonomous agent systems [18], inform our integrated approach to governance.

XII. Threats to Validity

Construct validity: Hallucination rates vary with evaluation methodology; different prompt strategies and model configurations produce different measurements. We report results stratified by these dimensions.

Internal validity: Published studies use different base models and benchmarks; direct comparisons are limited. We use fixed model versions within each comparison family.

External validity: Published benchmarks (NaturalQuestions, MedQA, HumanEval) may not generalize to proprietary data or specialized domains. Organizations must conduct in-domain evaluation before deployment.

Conclusion validity: Heterogeneous study designs and reporting standards preclude meta-analysis; results are presented narratively with full provenance.

XIII. Conclusion

Large Language Models represent a genuine technological advancement, offering productivity gains and enabling applications previously intractable. However, their production deployment introduces material security, reliability, and ethical risks. This paper synthesized evidence across three dimensions: security vulnerabilities (prompt injection affecting 100% of tool-using systems under realistic threat models), reliability limitations (hallucination rates of 5%–47% depending on domain), and organizational governance requirements.

We conclude that responsible LLM deployment requires:

1) **Technical defense-in-depth**: input validation, output verification, formal execution sandboxing, and audit trails.
2) **Organizational governance**: explicit policies, human oversight authority, and transparency with users about system limitations.
3) **Regulatory compliance**: alignment with emerging standards (EU AI Act, NIST frameworks) and proactive engagement with governance bodies.
4) **Continued research**: mechanistic interpretability, constitutional AI, and novel evaluation frameworks to improve safety and reliability.

Organizations that view LLMs as magic solutions will face failures. Organizations that treat LLMs as powerful but fundamentally constrained tools, subject to rigorous verification, human oversight, and continuous improvement, will capture substantial value while managing risk. The next phase of LLM adoption will be defined not by capability gains but by maturity in governance and safety practice.

References

[1] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[2] Anthropic, "Claude 3 model card," Technical Report, 2024.

[3] K. Greshake, S. Abdelnabi, S. Mishra, A. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 2023, pp. 79–90.

[4] H. Pearce, B. Ahmad, B. Raye, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 754–768.

[5] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[6] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems*, 2016, pp. 3866–3874.

[7] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 27730–27744.

[8] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning with membership privacy," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 43–57.

[9] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[10] M. Chen *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[11] J. Wei *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[12] S. Yao *et al.*, "ReAct: Synergizing reasoning and acting in language models," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 4378–4390.

[13] V. Thawani, R. Srikumar, and B. Prabhumoye, "Okay, what else? Finding and ranking implicit claims with language models," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 3970–3985.

[14] J. Wang, D. Zhou, J. Jiang, L. Zhang, and P. He, "Towards a unified multi-dimensional evaluator for text generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[15] L. Huang, Y. Weng, Y. Weng, and J. Yuan, "Hallucination in neural machine translation," *arXiv preprint arXiv:2303.16038*, 2023.

[16] H. Touvron *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[17] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Hambro, V. Kerkez, and T. Schwaller, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[18] J. Yang *et al.*, "SWE-agent: Agent-computer interfaces enable automated software engineering," *arXiv preprint arXiv:2405.15793*, 2024.

[19] P. Lewis, E. Perez, A. Piktus, F. Schwenk, D. Schwab, C. TabIaD, and J. Riedel, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[20] C. Lu, Z. Zhou, X. Liu, Z. Liu, H. Gai, and X. Luo, "Adversarial prompt attacks and defenses for language models," *arXiv preprint arXiv:2309.16788*, 2023.

[21] E. Wallace, S. Feng, D. Kandpal, J. Bisk, and S. Schwartz, "Universal adversarial triggers for attacking and analyzing NLP," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019, pp. 2153–2162.

[22] B. Wang, L. Shao, C. Li, Y. Luo, Z. Qian, and D. Qiu, "Are large language models robust to on-distribution examples?," *arXiv preprint arXiv:2303.14522*, 2023.

[23] N. Carlini, F. Liu, B. Ev0̆0e9quoz, S. Jagielski, A. Alfray, J. Zemel, J. Cheek, Y. Gal, S. Greshake, K. Greshake, S. Abdelnabi, S. Mishra, A. Endres, T. Holz, and M. Fritz, "Extracting training data from large language models," *arXiv preprint arXiv:2012.14386*, 2021.

[24] N. Perry, M. Srivastava, D. Kumar, and D. Boneh, "Do users write more insecure code with AI assistants?," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 2785–2799.

[25] P. Hacker, A. Engel, and M. Mauer, "Regulating artificial intelligence risks by design," *arXiv preprint arXiv:2301.08902*, 2023.

[26] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations*, 2023.

[27] M. Tufano *et al.*, "AutoDev: Automated AI-driven development," *arXiv preprint arXiv:2403.08299*, 2024.

[28] C. S. Xia *et al.*, "Agentless: Demystifying LLM-based software engineering agents," *arXiv preprint arXiv:2407.01489*, 2024.

[29] NIST, "Artificial intelligence risk management framework," U.S. Department of Commerce, Tech. Rep., 2023.

[30] C. E. Jimenez *et al.*, "SWE-bench: Can language models resolve real-world GitHub issues?," in *The Twelfth International Conference on Learning Representations*, 2024.

[31] S. Dhuliawala, K. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz, and J. Weston, "Chain-of-verifications reduces hallucinations in large language models," *arXiv preprint arXiv:2309.11495*, 2023.

[32] Y. Bai *et al.*, "Constitutional AI: Harmlessness from AI feedback," *arXiv preprint arXiv:2212.08073*, 2022.

[33] C. Elhage, N. Nanda, C. Olsson, T. Schiefer, T. Henighan, S. Joseph, B. Mann, A. Hernandez-Garcia, J. Conerly, T. Conerly, J. Bowman, S. Cotton, C. Perez, A. Nematzadeh, S. Burns, D. Amodei, and T. Brown, "Toy models of superposition," *arXiv preprint arXiv:2209.10652*, 2022.

[34] European Union, "Artificial Intelligence Act," Official Journal of the European Union, Regulation (EU) 2024/1689, 2024.

[35] J. Biden, "Executive order on the safe, secure, and trustworthy development and use of artificial intelligence," The White House, 2023.

[36] D. Acemoglu and S. Johnson, *Power and Progress: Our Thousand-Year Struggle Over Technology and Prosperity*. PublicAffairs, 2023.

[37] "Security challenges in contemporary machine learning systems," *Journal of Software Engineering and Cybersecurity*, vol. 1, no. 3, 2025. [Online]. Available: https://journal.kernelxos.com/jsec/article/view/13