A State-of-the-Art Survey on AI Systems for Multi-Task Productivity: From Large Language Models to Multi-Agent Systems

Your Name
Your Affiliation

***Abstract*—The rapid evolution of large language models (LLMs) has catalyzed the development of systems capable of performing multiple productive tasks. Early breakthroughs in language modeling paved the way for models to autonomously select and utilize external tools. Subsequent research focused on transforming these models into Large Action Models (LAMs) tailored for action execution. More recently, integrating LAMs within multi-agent systems (MASs) has emerged as a promising strategy for tackling complex, real-world tasks. This survey synthesizes historical progress, contemporary research, and future directions in the field—from LLMs using tools to MASs that achieve collaborative, autonomous task execution.**

***Index Terms*—LLMs, Toolformer, ToolLLM, Large Action Models, Multi-Agent Systems, Autonomous Agents, Collaborative AI**

I. Introduction

Large language models (LLMs) have redefined what is possible in artificial intelligence, demonstrating unprecedented capabilities in natural language understanding and generation. Despite their success, early LLMs were inherently limited to text-based outputs and struggled with tasks requiring real-world action. Recent work has demonstrated that LLMs can be extended to identify and call appropriate tools, effectively bridging the gap between language and action. This survey reviews the evolution from tool-using LLMs to specialized Large Action Models (LAMs), and ultimately to multi-agent systems that harness these capabilities for collaborative, productive task execution.

In the following sections, we outline:

- **How we got here:** A brief history of LLMs' transformation through tool usage.
- **Where we are now:** The emergence of LAMs and their integration into multi-agent frameworks.
- **Where we could go next:** Challenges faced by current systems and potential directions for future research.

II. Literature Review

From Language Models to Tool-Using Agents

Advancements in artificial intelligence have accelerated in recent years, particularly with the introduction of the transformer architecture. This innovation enabled commercial LLM-supported products and significantly impacted society's understanding of AI. From a product engineering perspective, where user interface design is paramount and team value comes from well-crafted experiences, the chat interface has been widely embraced. This rapid adoption is no coincidence—our first interactions with LLMs occurred through interfaces that create the impression of conversing with another person, resulting in a natural experience that increased user adoption.

Initially, the value users derived from these interfaces was largely limited to text generation. As interactions became multimodal, the term "Generative AI" become popular to describe these systems. This term reflects both their architectural approach to generating outputs and their perceived intelligence and creativity—their ability to create new ideas, images, and sounds. This framing helps us understand phenomena like "hallucinations" and other characteristics of these AI systems.

In brief, contrary to science fiction's predictions, we have developed a form of artificial intelligence that excels more at creative tasks than logical ones. This observation sets the foundation for our literature review: while current AI shows immense potential, it still lacks the reliability needed for complex task execution.

In this research exercise, we identify a trend of evolution that expands around 3 years, i.e., [1], [2]. Researchers noticed the potential of the LLMs to be used for more than just text generation, as they display excellent ability to understand and produce structured output, there was a possibility of integrate with existing software artifacts already in place, such as APIs, SDKs, and any other programmable interface.

API's where one of the most natural places to start with. Toolformer (Schick et al., 2023) [1] represents a breakthrough in addressing LLM limitations through tool integration, enabling models to autonomously determine when external tools are needed, select appropriate ones, and incorporate their outputs into ongoing language generation. Its distinctive self-supervised learning methodology implements a streamlined three-step process that samples potential API calls using in-context learning, executes these calls to obtain results, and filters them based on perplexity reduction. This approach creates a more adaptive system capable of determining which tools benefit specific contexts without extensive human annotation.

Expanding this line of research, ToolLLM (Qin et al., 2023) [2] introduces a comprehensive framework for training models to effectively utilize external tools through API calls. It contributes three key innovations: a systematic methodology for converting conventional dialogue datasets into tool-use examples through automated annotation; ToolBench, a diverse dataset spanning over 16,000 real-world scenarios across various domains; and a training approach enabling models to make effective API calls without human intervention. Perfor-

mance evaluations demonstrate that tool-augmented models significantly outperform standard approaches, particularly in generating syntactically correct API calls, tool selection, and output integration.

These complementary innovations mark a crucial first step in improving the reliability of AI systems for performing specific tasks by establishing a new paradigm where language models can handle tasks beyond linguistics to perform useful, real-world operations. By maintaining sophisticated language understanding while gaining access to specialized functionality, tool-augmented systems offer a cost-effective approach to enhancing model capabilities without increasing model size, potentially transforming how AI assistants support human productivity across diverse domains.

Large Action Models (LAMs)

Tool-using LLMs introduced a paradigm shift in the way we perceive the functionality of AI systems, the main idea was: "Now we have a system that can interact with our world", in the context of our modern digital productivity tools. Taking a step further in the evolution of an LLM using tools, our research found the concept of "Large Action Models" which introduces a new action layer that positions AI systems at the same level of interaction as modern application layers. This advancement has produced systems capable of performing actions in dynamic environments. Although LAMs did not require significant architectural changes to LLMs, the name effectively captured the new direction of applied AI. We now proceed to describe this process.

Developing and Training Large Action Models: The goal is to transform a general-purpose LLM into a specialized action execution system. This transformation requires designing a dedicated pipeline. Wang et al. [3] describe this process in detail. Understanding this pipeline is important not only for this paper but also because these methods can be applied broadly to similar objectives, enriching our understanding for future development. The process consists of several interconnected stages:

Data Collection and Preparation: As most of AI related work, everything starts with gathering and curating task-specific data. This initial phase follows a two-phase approach:

- **Task-Plan Data Collection**: This involves collecting user requests and corresponding step-by-step plans. Sources include application documentation, WikiHow articles, and historical search queries. Each entry typically contains a task description and a detailed plan outlining the steps required to accomplish it.
- **Task-Action Data Collection**: This phase converts task-plan data into executable task-action data. Each plan step is transformed into concrete, actionable instructions that can be directly executed in the target environment. This process includes instantiation (adding specific operational details), execution validation, and evaluation to ensure correctness.

For example, a typical entry might look like:

```
{
  "task_id": "powerpoint_task_001",
  "task": "Create a slide based on draft.docx"
    ,
  "plan": [
    "1. Open the draft.docx and read the
    content.",
    "2. Create a new PowerPoint file.",
    "3. For page 1, add ..."
  ],
  "actions": [
    {
      "step": "open the document",
      "function": "open",
      "args": {"file_path": "draft.docx"}
    }
  ]
}
```

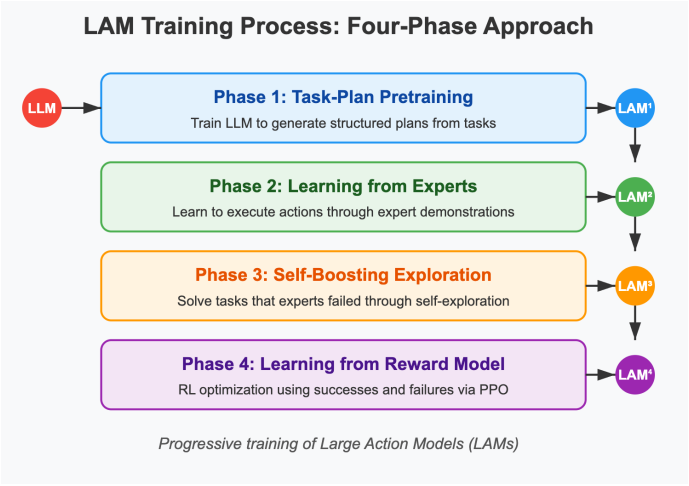

Fig. 1. Progressive four-phase training approach for transforming a general LLM into a specialized LAM.

Transforming an LLM into a LAM, a Four-Phase Approach: Wang et al. [3] propose a systematic, four-phase training process that progressively builds capabilities from planning to action execution. As our LLM model progresses through these phases, it becomes a LAM:

1) **Task-Plan Pretraining**: In this foundational phase, the model (LAM^1) learns to generate structured, step-by-step plans for completing tasks. Using supervised fine-tuning with task-plan pairs, the model develops the ability to break tasks into logical sequences. At this stage, the model can generate coherent plans but lacks execution capabilities.
2) **Learning from Experts**: In the second phase, the model (LAM^2) learns to translate plans into concrete, executable actions through expert demonstrations. Training uses state-action pairs where each state includes the current UI environment and task context, and each action specifies which control to interact with and how. This phase grounds the model's reasoning in real application environments.
3) **Self-Boosting Exploration**: In the third phase, the system learns to improve itself. LAM^2 attempts to solve

tasks that were unsuccessful in previous stages. Using a ReAct mechanism, [4], it interacts with the environment and explores alternative strategies for these challenging tasks. When successful, it generates new solutions. These solutions are then combined with the original expert examples to create an enhanced training dataset for LAM[3].

4) **Learning from a Reward Model**: The final phase introduces reinforcement learning, enabling the model to learn from both successes and failures. First, a reward model is trained on both successful and failed trajectories, assigning positive values to successful steps and negative values to failures. Then, LAM[4] is fine-tuned using Proximal Policy Optimization (PPO) on previously failed trajectories, guided by the reward model's feedback. This approach enhances the model's decision-making abilities in complex scenarios.

Each step in the training process builds on the one before it. This creates AI systems that get better and better at four key things: Planning out steps to solve problems, learning from expert examples, figuring things out on their own, getting better through trial and error.

Integration with Agents: Once the trained is complete, it is neccessary to close the breach between strategy and execution. In few words, we have a very capable model to engage in planning to resolve tasks, but it's necessary to add interaction with the environment where it will operate. We introduce the concept of Agents—though their meaning and specific definition are still evolving—which, in the context of this paper, represent a layer of interaction between LAMs and the environment. LAMs must be integrated into an agent framework to interact with real-world environments. To make LAMs work in the real world, they need four main parts working together:

1) A way to see and understand what's happening in their environment, usually through special software interfaces (UI Automation APIs)
2) A system that turns the model's decisions into actual actions in the environment
3) A memory system that keeps track of what actions were taken and what plans were made, helping the system stay organized and consistent
4) A feedback system that watches how the environment responds to actions, helping the system learn and adjust to changes

Put simply, the LAM functions as the brain while the Agents serve as the body, complete with hands that can manipulate tools and execute plans.

Evaluation: Evaluation is a critical component of any DL/ML system, serving as the feedback loop for system improvement. While Wang and colleagues [3] established clear testing methodologies for comparing different versions and ensuring proper functionality across various situations, the current evaluation scope remains limited to a 435-task benchmark focused on Microsoft Word tasks. Nevertheless, their comprehensive development pipeline demonstrates LAMs' ability to translate user intentions into meaningful actions within specific operational contexts, representing a significant advancement over traditional text-generation models by enabling the execution of complex, multi-step tasks in real-world environments—though more extensive testing across diverse domains is still needed.

Applications and Impact: LAMs enable a wide range of applications previously challenging for traditional LLMs, including:

- Autonomous interaction with software applications and operating systems
- Control of physical devices and robotic systems
- Complex task automation in specialized domains like healthcare or finance
- Multi-step problem-solving that requires environmental awareness

These capabilities mark a significant advancement toward artificial general intelligence (AGI), enabling AI systems to automate tasks that were previously only possible with human intervention.

LAMs thus represent a paradigm shift—from language generation to action execution—allowing AI systems to automate complex processes with minimal human intervention, substantially expanding their practical utility across numerous domains.

Multi-Agent Systems

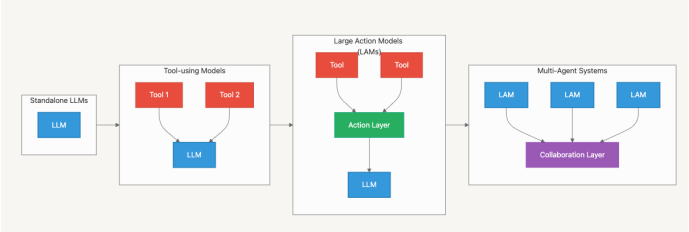

Fig. 2. Evolution of AI systems from standalone LLMs to multi-agent systems.

At this point we hope the idea progression start to make more sense. We began with general purpose LLMs learning to use tools, then with a extra phase of fine tunning they specialized in strategic habilities and along with agents, they become more capable of resolving tasks. We now introduce the concept of multi-agent systems. The integration of Large Action Models (LAMs) into multi-agent systems (MASs) represents the next frontier in AI system design. Mutliple agents work togheter to accomplish tasks that exceed the capability of any single agent. This collaborative approach fundamentally transforms how AI operates by shifting from isolated models toward interconnected systems that can distribute complex problems. If the challenge in LAMs was to design specific training to specialize our LLM, now the challenge for multi agent systems is to stablish roboust methods for a collaboration layer, which should enable them to communicate, coordinate, and collectively tackle complex, multi-step challenges.

According with [5], the collaboration mechanisms in these systems can take various forms. In cooperative scenarios, agents align their individual objectives with shared goals, often specializing in complementary roles to enhance efficiency. This approach has proven successful in applications ranging from question answering to collaborative programming, where different agents may handle research, planning, execution, and evaluation tasks. By contrast, competitive collaboration occurs when agents operate with potentially conflicting objectives, which can drive innovation and robustness through evolutionary pressure. Such competition-based approaches have shown benefits in debate frameworks and strategic gaming environments, where contradictory perspectives often yield more nuanced and comprehensive solutions. A hybrid approach—coopetition—enables agents to collaborate on certain tasks while competing on others, facilitating negotiation and compromise in complex, multi-objective problems.

The communication structure of these multi-agent systems further defines how information flows between collaborating agents. In centralized architectures, communication follows a hub-and-spoke model where interactions flow through a central coordinator agent that manages task allocation and synthesizes results. Decentralized structures enable peer-to-peer interactions without centralized control, offering greater resilience against single points of failure but requiring more sophisticated coordination mechanisms. Hierarchical arrangements create layered structures with defined authority levels, balancing control and flexibility across different tiers of agents with varying specializations.

The coordination protocols governing these interactions range from rule-based approaches with predefined interaction guidelines to role-based systems that divide labor according to agent specialization, and model-based frameworks that adapt probabilistically to environmental uncertainty. Such protocols ensure that agent collaborations remain coherent, efficient, and aligned with overall system objectives.

By leveraging LAMs within multi-agent frameworks, these systems gain several critical advantages. Complex tasks can be decomposed and allocated across specialized agents, with each focusing on subtasks aligned with its particular strengths. Resources can be dynamically allocated based on agent expertise and availability, allowing the system to adapt to changing requirements. The inherent redundancy in agent capabilities ensures resilience when individual components fail, while the collective intelligence that emerges from these collaborations often exceeds what any single agent could achieve independently. Moreover, these systems can scale by integrating new agents with specialized functions as requirements evolve.

This integration marks a significant leap in AI system design, enabling sophisticated real-world applications where agents interact, negotiate, and cooperate autonomously. From software development teams where different agents handle specification, coding, testing, and documentation, to creative content generation where specialized agents contribute different aspects of the creative process, to research assistance and complex decision-making scenarios, LAM-powered multi-agent systems are expanding the boundaries of what AI can accomplish.

Despite their promise, these systems face important challenges that ongoing research must address. Coordination complexity increases substantially as more agents join the system, requiring sophisticated mechanisms to ensure coherent task allocation and planning. Communication overhead must be carefully balanced against the benefits of collaboration to maintain computational efficiency. Error propagation presents another concern, as individual agent mistakes can potentially cascade through the system without proper safeguards. Finally, comprehensive evaluation metrics and benchmarks for assessing multi-agent performance remain underdeveloped compared to those for single-agent systems.

As research in this field advances, addressing these challenges will be crucial for realizing the full potential of multi-agent systems in productive AI applications. The trajectory from language models to action-capable agents to collaborative multi-agent systems represents not merely an incremental improvement but a fundamental paradigm shift in how AI systems can perceive, reason, and act in the world.

Applications and Use Cases

Recent advances in AI systems have led to practical applications across various domains. The evolution from LLMs to LAMs and beyond has created a rich ecosystem of specialized models with distinct capabilities: Large Action Models (LAMs) LAMs have emerged as powerful tools for structured decision-making environments. These models typically consist of a state encoder for processing structured input, a policy network for generating actions, and a value network to estimate expected rewards. Notable examples include:

xLAM – A Family of Large Action Models: The xLAM series, pioneered by Salesforce, exemplifies how LAMs can empower AI agent systems [6]. These models, spanning various sizes and architectures, have demonstrated excellence in executing real-world tasks, including tool use and interactive operations. Game AI Systems: DeepMind's AlphaGo was the first AI to defeat a human champion at Go, utilizing deep neural networks with Monte Carlo Tree Search (MCTS) to evaluate board states and optimize long-term strategies [7]. Complex Game Environments: OpenAI Five leveraged self-play and Proximal Policy Optimization to master the multiplayer game Dota 2, demonstrating LAMs' ability to handle complex, dynamic environments with multiple agents [9]. Autonomous Driving: Companies like Waymo employ convolutional neural networks to interpret sensor data (LIDAR, cameras) alongside reinforcement learning-based planning frameworks to produce driving behaviors [10]. Similarly, Tesla's Full Self-Driving system utilizes transformer-based neural networks that process multiple input streams simultaneously to make real-time driving decisions [11].

III. Future Directions

The evolution from isolated language models to integrated multi-agent systems represents a pivotal transformation in artificial intelligence capabilities. While impressive strides

have been made, the journey ahead presents several critical challenges that must be addressed to unlock the full potential of these systems. As we envision the future landscape of AI agents, we see a progressive path where advances in several interconnected areas will collectively drive the field forward.

The foundation of effective multi-agent systems begins with sophisticated task planning and coordination mechanisms. Current approaches struggle to optimally distribute responsibilities across agents with specialized capabilities, particularly as system complexity increases. Future research must develop dynamic allocation algorithms that understand not only what each agent can do but when it should do it. The introduction of structured debate loops, where agents iteratively refine intermediate results through collaborative reasoning, shows particular promise. These approaches, potentially guided by game-theoretic frameworks like Stackelberg or Nash equilibria, could dramatically enhance collective problem-solving capabilities.

As these coordination mechanisms evolve, they must be supported by increasingly sophisticated memory and context management systems. Unlike single-agent frameworks, multi-agent systems must juggle multiple layers of context—from agent-specific knowledge to shared consensus information. This introduces unique challenges in maintaining contextual alignment across agents while preserving the integrity of private information. The development of hierarchical memory architectures with appropriate access controls, along with advanced episodic memory systems that learn from past collaborative interactions, will prove crucial for coherent system operation.

The bridges between these islands of specialized intelligence—communication frameworks—require equal attention. Current systems often suffer from inefficient information exchange that creates bottlenecks as agent populations grow. Standardized protocols that enable seamless integration of diverse agent types, coupled with sophisticated negotiation and consensus-building mechanisms, will be essential for scaling these systems beyond simple demonstrations to robust, production-ready applications.

As multi-agent systems grow in complexity and scope, concerns about reliability and resilience become increasingly pressing. Methods for preventing error propagation, ensuring graceful degradation when individual agents fail, and dynamically integrating new agents into operational systems will be necessary to maintain trust in these increasingly autonomous frameworks. This technical robustness must be paired with comprehensive evaluation methodologies—an area where current research falls notably short. The field urgently needs metrics and benchmarks specifically designed to assess collaborative performance and emergent system capabilities, not just individual agent effectiveness.

The application of these advances to specific domains presents both unique challenges and opportunities. The integration of multi-agent systems with blockchain technologies, as highlighted in recent research [7], offers promising directions for enhancing distributed system capabilities. Similarly, embedding these systems within enterprise software ecosystems or bridging digital reasoning with physical action through robotics presents domain-specific hurdles that will require specialized research attention.

Addressing these interconnected challenges demands collaborative efforts spanning artificial intelligence, distributed systems, human-computer interaction, and domain expertise. As researchers tackle these problems, we anticipate a gradual transition from today's experimental prototypes to tomorrow's production-ready systems—systems capable of reliably executing complex, collaborative tasks across diverse real-world environments. This transition will not only advance technological capabilities but potentially redefine how we conceptualize the boundaries between human and artificial intelligence in productive work.

IV. Conclusion

The evolution from large language models to multi-agent systems represents a fundamental transformation in AI capabilities. This journey—from text generation to autonomous action execution—has progressed through distinct stages: tool-using LLMs that identified when to leverage external functionalities; specialized Large Action Models (LAMs) trained to execute complex task sequences; and collaborative multi-agent systems where specialized agents work together to solve problems beyond individual capabilities. This progression has expanded AI's practical utility across domains, enabling sophisticated automation previously requiring human intervention. However, significant challenges remain in coordination mechanisms, communication protocols, error resilience, and evaluation methodologies. Future research must focus on developing robust frameworks for agent collaboration, standardized communication, and comprehensive performance metrics tailored for multi-agent interactions. As these systems mature, we anticipate a shift from experimental prototypes to production-ready applications capable of executing complex, collaborative tasks in diverse real-world environments. This transition will not only advance technological capabilities but may fundamentally redefine human-AI collaboration in productive work across industries—from software development to creative content generation to strategic decision-making.

IV. References

[1] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, Feb 2023, [Online]. Available: https://arxiv.org/abs/2302.04761. [Accessed: Mar. 14, 2025].

[2] Y. Qin, S. Liang, H. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, D. Li, Z. Liu, and M. Sun, "Toolllm: Facilitating large language models to master 16000+ real-world apis," *arXiv preprint arXiv:2307.16789*, Oct 2023, [Online]. Available: https://arxiv.org/abs/2307.16789. [Accessed: Mar. 14, 2025].

[3] L. Wang, F. Yang, C. Zhang, J. Lu, J. Qian, S. He, P. Zhao, B. Qiao, R. Huang, S. Qin, Q. Su, J. Ye, Y. Zhang, J.-G. Lou, Q. Lin, S. Rajmohan, D. Zhang, and Q. Zhang, "Large action models: From inception to implementation," *arXiv preprint arXiv:2412.10047*, Jan 2025, [Online]. Available: https://arxiv.org/abs/2412.10047. [Accessed: Mar. 14, 2025].

[4] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, Oct 2022, [Online]. Available: https://arxiv.org/pdf/2210.03629. [Accessed: Mar. 14, 2025].

[5] K.-T. Tran, D. Dao, M.-D. Nguyen, Q.-V. Pham, B. O'Sullivan, and H. D. Nguyen, "Multi-agent collaboration mechanisms: A survey of llms," *arXiv preprint arXiv:2501.06322*, Jan 2025, [Online]. Available: https://arxiv.org/abs/2501.06322. [Accessed: Mar. 14, 2025].

[6] J. Zhang, T. Lan, M. Zhu, Z. Liu, T. Hoang, S. Kokane, W. Yao, J. Tan, A. Prabhakar, H. Chen, Z. Liu, Y. Feng, T. Awalgaonkar, R. Murthy, E. Hu, Z. Chen, R. Xu, J. C. Niebles, S. Heinecke, H. Wang, S. Savarese, and C. Xiong, "xlam: A family of large action models to empower ai agent systems," *arXiv preprint arXiv:2409.03215*, Sep 2024, [Online]. Available: https://arxiv.org/abs/2409.03215. [Accessed: Mar. 14, 2025].

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016, [Online]. Available: https://github.com/tpn/pdfs/blob/master/Mastering%20the%20Game%20of%20Go%20with%20Deep%20Neural%20Networks%20and%20Tree%20Search.pdf. [Accessed: Mar. 14, 2025].