Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2, 73-90
2026
Publisher: Learning Gate
DOI: 10.55214/2576-8484.v10i2.11983

Approaches and emerging trends in multi-agent autonomous AI systems for education innovation in Vietnam

Cam Tran Ai[1], Hung Tran Van[2*], Trung Tran[3], Hien Lam Thanh[4]
[1]Intracom University, Vietnam; tranaicam75@gmail.com (C.T.A.).
[2]The University of Danang, University of Science and Education, Danang, Vietnam; tvhung@ued.udn.vn (H.T.V.).
[3]Vietnam Academy for Ethnic Minorities, Vietnam; trungt1978@gmail.com (T.T.).
[4]Lac Hong University (LHU), Bien Hoa City, Dong Nai Province, Vietnam; lthien@lhu.edu.vn (H.L.T.).

Abstract: Agentic artificial intelligence (AI) systems that act as autonomous agents are rapidly evolving due to the explosion of large language models (LLMs) [1]. This paper presents a comprehensive analysis of major approaches in the field of Agentic AI, including the large visual language model (LVLM), the React and Plan-and-Execute agent architectures, the smolagents library with a "coding-first" orientation, tool invocation techniques that extend the capabilities of LLMs, the visual Agentic AI model with multi-agent coordination capabilities, as well as scientific agent systems such as AI Scientist and the AgentRxiv collaboration platform. We analyze the characteristics of each approach, including representation models, advantages, limitations, and integration capabilities, for building intelligent agent systems that aim for AGI. The paper proposes an integrated scheme that leverages achievements from multimodal capabilities, multistep reasoning and planning, multi-agent coordination, and research automation, laying the foundation for a new generation of autonomous AI agents. Finally, we discuss the potential applications of Agentic AI in the context of Vietnam, especially in education, scientific research, and technology development, and provide recommendations for domestic developers and researchers.

Keywords: *AI agents, Artificial general, Education, Large language models, Large vision-language models, Multi-agent, LLM tools, Plan-and-execute, ReAct.*

1. Introduction

The advent of LLMs such as GPT-3/4 has revolutionized natural language processing, providing superior text understanding and generation capabilities [2]. However, to effectively apply LLMs to complex real-world tasks, we need more than a simple response model. The concept of Agentic AI is proposed to refer to AI systems that act as autonomous agents (AI agents), that is, they can automatically receive tasks, plan their steps, interact with the environment (e.g., search for information, call external tools, analyze data), and adjust their actions to complete the set goals [3].

Since the beginning of 2023, the AI community has witnessed the emergence of prominent projects, such as AutoGPT, BabyAGI, and AgentGPT, demonstrating the potential of LLMs when they are "empowered" to act continuously without human intervention [4]. These AI agents are expected to undertake complex, multi-step processes – such as automating information retrieval and summarization, programming, or even supporting scientific research. In Vietnam, interest in LLM and Agentic AI has also increased sharply after the launch of ChatGPT. Many domestic organizations and research groups have quickly adapted to the trend, developed Vietnamese language models, and organized in-depth courses (for example, the AI Vietnam AI Course 2024 series) to equip participants with the knowledge needed to build AI agents. This context presents an essential premise for the Vietnamese scientific and educational community to grasp and contribute to the global wave of Agentic AI.

History: Received: 14 November 2025; Revised: 5 January 2026; Accepted: 9 January 2026; Published: 3 February 2026
* Correspondence: tvhung@ued.udn.vn

In this paper, we review the typical technological directions shaping the Agentic AI ecosystem. Part 2 introduces (i) LVLMs, a multi-modal model combining vision and language; (ii) the ReAct agent architecture and its Plan-and-Execute variant, enabling LLMs to reason and act in multiple steps; (iii) the SmolAgents lightweight library with a "code-first" philosophy in agent design; (iv) the tool calling method, a bridge allowing LLMs to call external tools; (v) the Visual Agentic AI model, which uses a multi-agent supervisor architecture to coordinate multiple specialized agents; (vi) the AI Scientist automated scientific research agent framework; and (vii) the AgentRxiv platform that supports agents in collaborating on knowledge sharing. Section 3 analyzes the main advantages and limitations of each approach and proposes an integration scheme that enables these technologies to converge, building increasingly comprehensive AI agents that move closer to the goal of AGI [5]. Section 4 discusses prospects for applying Agentic AI in education, research, and industry in Vietnam. Finally, Section 5 summarizes future trends and offers recommendations for domestic developers and researchers to catch up and contribute to this potential field.

2. Primary Approaches in Agentic AI

2.1. LVLM Models

LVLMs are deep learning models designed to process and understand both visual and natural language data simultaneously [6]. Instead of separating the two fields of computer vision and natural language processing, LVLMs combine them into a single system. By leveraging the representational power of LLMs on language, LVLM models can annotate images, answer questions about visual content, and perform many other multimodal tasks flexibly [7].

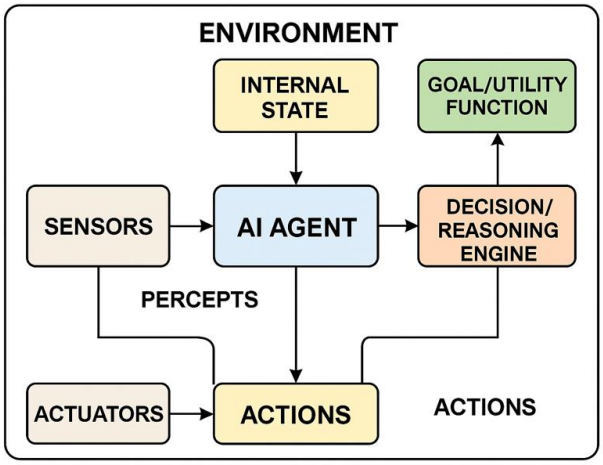

Figure 1.
LVLM's multitasking capabilities.

Figure 1 illustrates the multitasking capabilities of LVLMs: the model can locate objects in images, perform image segmentation without prior training, and answer questions about images, for example, identifying the breed of a cat without being specifically trained for that task [8]. This ability to "understand" both the visual and textual worlds opens up rich AI applications and lays the foundation for moving toward building AGIs that can perform any human-like intellectual task [9]. However, LVLMs still face several challenges. On one hand, large-scale models require substantial computational resources and training data, which are difficult for resource-constrained research teams [6]. On the other hand, the accuracy of LVLMs can degrade significantly when faced with data or questions outside the original training data distribution [10]. For example, an LVLM trained primarily in natural images may struggle when asked to analyze specialized medical images. To overcome this, supervised fine-tuning on the target task is being pursued to fine-tune LVLMs for specific tasks [11]. However, overall,

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

LVLMs are still considered an important step toward future multi-modal AGI systems, where an AI agent can both "see" the world, "understand," and "act" through language.

2.2. ReAct and Plan-and-Execute Architectures

A significant challenge in building LLM-based AI agents is enabling the model to reason in multiple steps and act sequentially to achieve complex goals. Two typical agent architectures addressing this are ReAct and Plan-and-Execute, which currently attract much attention in the community [12]. Both belong to the class of reasoning agents, agents capable of reasoning and making decisions in sequence, but their approaches differ slightly. ReAct is a method that allows LLMs to repeat the "Think -> Act -> Observe" cycle until a result is achieved [13]. Specifically, the ReAct agent operates in a loop: the model analyzes the request and generates a stream of thoughts in the form of an internal reasoning chain, from which it decides to perform a specific action [14] (e.g., call a tool, perform a calculation, query information); after the action, the model receives observations about the results of that action (e.g., the content returned from the tool), updates its state, and continues the loop [15]. This process repeats N steps until the task is completed, and the model gives a final answer. Figure 2 illustrates the ReAct cycle: the agent starts with a user request, generates thoughts to plan, then executes the corresponding action through a tool, receives the observation results, and continues to think for the next step – this process is repeated many times dynamically until the task is completed [16]. The advantage of ReAct is its adaptability and flexibility: the model can adjust its strategy based on new information gained after each action, much like humans do when testing and refining their reasoning while solving a problem [17]. As a result, ReAct is effective in situations where it is not possible to have a fixed plan at the beginning or where it is necessary to react flexibly to the environment.

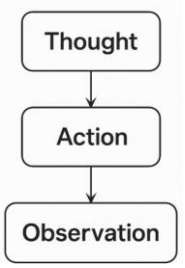

Figure 2.
ReAct agent cycle.

In contrast to the adaptive iteration of ReAct, the Plan-and-Execute architecture emphasizes sequential planning in an organized manner [18]. As the name suggests, this approach separates the task-solving process into two distinct phases: planning and execution [19]. Specifically, in the first phase, the LLM agent will carefully analyze the requirements and develop a detailed, overall plan of the steps to be taken to achieve the goal [20]. This plan is like a pre-drawn action scenario. Then, in the second phase, the agent will sequentially execute each step according to the proposed plan until the task is completed. The primary advantage of Plan-and-Execute is that the agent has a comprehensive view of the entire task from the outset, which helps prevent missing important steps and ensures the logical structure of the solution [21]. This method is beneficial for complex problems that require deep and systematic reasoning, where considering the whole in advance is more effective than working on small, undirected steps.

However, Plan-and-Execute also has limitations compared to ReAct. Since the entire plan is created before the action, the model may encounter difficulties if the environment changes or new information is unexpectedly introduced, in which case the original plan may no longer be optimal. ReAct, on the other hand, does not have a fixed plan from the beginning, so it is highly flexible in adjusting to the actual situation [22]. In short, ReAct is an adaptive, reflexive approach, while Plan-and-Execute is a

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

structured, sequential approach. The choice of architecture depends on the nature of the task: simple tasks or dynamic environments may be suitable for ReAct, while complex tasks, where the main steps can be predicted in advance, benefit from Plan-and-Execute [23]. Some systems attempt to combine both advantages – for example, by building agents that start with framework planning but still allow for flexible adjustments during execution or by coordinating multiple agents where one agent plans and another executes [24].

In the AI community, ReAct and Plan-and-Execute are implemented through various tools and libraries. A typical example is the use of LangChain/LangGraph to model the inference cycle of an agent. The LangGraph library (based on LangChain) enables representing the task flow of LLMs as a looped state graph, which is well-suited to realizing both ReAct and Plan-and-Execute architectures. Specifically, LangGraph provides components such as ToolNode that allow the agent to invoke tools or StateGraph to manage the state through steps [25]. Figure 3 below illustrates how a ReAct agent can be configured using LangGraph: each Inference–Act–Observe loop is represented as a journey through state nodes. LangGraph helps to track and repeat this cycle in a controlled manner.

2.3. Smolagents: Code-first Agent Design

While architectures like ReAct focus on inference logic, another practical aspect of building AI agents is the programming and deployment tooling. Most libraries, such as LangChain, provide multiple layers of abstraction and integration; however, they can sometimes be quite complex and cumbersome for simple applications [26]. Responding to the need for a more lightweight and intuitive solution, the Hugging Face team released Collagens, a small Python library for building AI agents. As the name suggests, the philosophy of smolAgents is to keep things as simple as possible. The library implements the agent logic in approximately 1,000 lines of Python code, making the code easy to read, understand, and debug, thereby avoiding unnecessary layers of abstraction.

The unique feature of smolagents is the "code-first" approach: instead of having LLM generate actions in text or JSON and then execute them, smolagents allows the agent to write programming code (Python) directly to call the tool and perform the action [27]. In other words, the agent's thinking is expressed directly in code, just like the agent itself programs the steps to solve its problem. This approach has the advantage of being clear and transparent and, at the same time, makes the most of the code generation capabilities of the large language model [28]. Smolagents supports two main types of agents: CodeAgent and ToolCallingAgent. Depending on the problem, the developer can choose to represent the action in code or JSON accordingly. Internally, smolagents implement the agent operation mechanism based on the React framework mentioned above. Each agent inherits from the MultiStepAgent class and performs a "reason-act-observe" loop through multiple steps, functioning as a state machine. At each step, the smolagents agent will: (i) Think, call the LLM to generate a plan/action in code; (ii) Act, execute that code, which may include calling tools or calculations; (iii) Observe, get the return result (function result or tool output) and pass it back to the LLM in the next thinking step. This process repeats until the final answer is received or the number of steps is reached. Thanks to being built on Hugging Face Hub, smolagents can also easily integrate existing models and share tools through the Gradio Spaces interface.

To better visualize this, let us consider a simple example: building an automated news summarization agent. With smolagents, the programmer can define a set of "tools," such as (1) a search tool to search for news via API, (2) a scrape tool to get article content, and (3) a summarize tool to summarize the text. Then, using the CodeAgent class, we let the agent write Python code to call these tools in turn: first, use search_tool to find the latest AI news headlines, then scrape_tool to retrieve the content of each news article, and finally, summarize_tool to create a concise summary. All of this logic is automatically generated and executed by the agent in sequence. The detailed process is recorded by smolagents, making it easy for users to track and debug [29].

Overall, smolagents provide a flexible and powerful solution for building complete AI agents from simple tasks such as querying and synthesizing information to more complex problems requiring multi-

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

step interactions. The focus on code helps reduce system complexity while leveraging the automatic programming capabilities of LLMs. In Vietnam, small agents can be helpful in rapidly developing specialized virtual assistants when programmers can easily customize the agent logic through lightweight source code instead of having to train the model from scratch. Of course, the downside of the "code-first" approach is that it requires LLMs to be truly reliable in generating code; this depends on the quality of the model and thorough testing. However, with the progress of LLMs today, smolagents are proof that building AI agents does not have to be complicated but can be "lightweight and efficient."

2.4. Extending LLM Capabilities with External Tools

An inherent limitation of pure LLM models is that they cannot directly interact with the outside world or access new information after training [30]. LLMs are trained on static datasets, so they are unaware of events that occurred after the training data point, nor can they look up or perform precise calculations on their own, relying solely on statistical memory. To overcome this, researchers and developers have developed a solution called Tool Calling, which enables LLM models to call external tools or functions as part of the response process [31]. Simply put, when encountering a question that exceeds its internal knowledge or skills, LLM can "request" a suitable tool, then take the results returned by the tool to synthesize into the final answer.

The tool-calling mechanism is often integrated as an interactive loop between the LLM and the environment, similar to the ReAct architecture presented by Gim et al. [32]. Figure 4 illustrates the process: when a user asks a question, instead of answering immediately, the model first determines whether to utilize an external tool [33]. If necessary, it will stop generating the answer and switch to calling an AI agent that acts as an intermediary to execute the tool [34]. The tool is run and returns the result. The LLM then takes that result, incorporates it into the context, and generates a final response to the user. The entire system works in a cycle: Think (determine what tool is needed) → Invoke tool → Observe result → Respond. This significantly expands the capabilities of the LLM, transforming it from a "read and speak" agent to a genuine agent that can interact with reality [35].

A concrete example: suppose a user asks the chatbot, "What is the weather like in Paris today?" A regular LLM would struggle to answer correctly if it did not have access to the actual weather data. However, with tool calling, the model can recognize that this is a weather question and decide to call a weather API. It will make a function call, such as get_weather, which is sent to the tool agent. This tool, which can be pre-programmed or coded by the LLM, will execute by querying a weather service and return a result, for example: "Temperature 18°C, light rain." The LLM receives this information and finally replies to the user: "It is light rain in Paris today, around 18 degrees C." Thanks to tool calling, the chatbot has provided an up-to-date and accurate answer, something that the language model itself could not do if it relied only on old data. Tool calling has been integrated into large systems, typically ChatGPT (GPT-4), with Plugins and a function calling API. Technically, there are two common ways to implement it: (1) Direct instruction, LLM is provided with special "prompts" to output the tool calling format if needed, then a dispatcher reads that JSON and executes the corresponding function; (2) Code generation, like smolagents or GitHub Copilot, the model generates code to call the tool and then runs that code. Either way, the common point is that LLM plays a central role in determining what to do, while external tools handle specialized tasks.

Why is tool calling important? Because it overcomes the limitations of LLM in many ways:

- First, the model can access the latest information instead of being limited to static training knowledge [36].
- Second, the model can perform precise computational tasks by calling computers or code instead of relying on fallible inference.
- Third, the model can interact with other systems, paving the way for many automation applications.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

In Vietnam, the ability to combine LLM with domestic tools will create versatile AI assistants that serve users better. For example, an educational virtual assistant can look up learning materials from an electronic library, solve math problems by calling the CAS system, or support multilingual translation by integrating a machine translation model. Although implementing tool calling requires safety considerations, with appropriate guidelines and sandboxes, this is undoubtedly an indispensable component of modern agentic AI systems. It can be said that tool calling is the bridge that transforms a text-generation model into a knowledgeable agent that can observe and influence the world rather than just "sitting and talking."

2.5. Visual Agentic AI and Supervisor Architecture

Another limitation of deploying a single AI agent system is the risk of overloading it by requiring it to take on a wide range of tasks and skills. In reality, no single agent can excel at strategic planning, insight searching, digital data analysis, image recognition, and other tasks simultaneously [37]. Furthermore, when integrating multiple tools, a single model must select the right tool from dozens of options, which increases complexity and the risk of errors [38]. To overcome this, multi-agent architectures have been proposed as a possible solution: instead of a single, all-powerful agent, the system is divided into independent agents, each specialized in a specific task, which can communicate with one another or be coordinated by a supervisory agent [39]. There are many structures for organizing agents in multi-agent systems. Figure 3 below summarizes six common agent architectures, ranging from simple to complex.

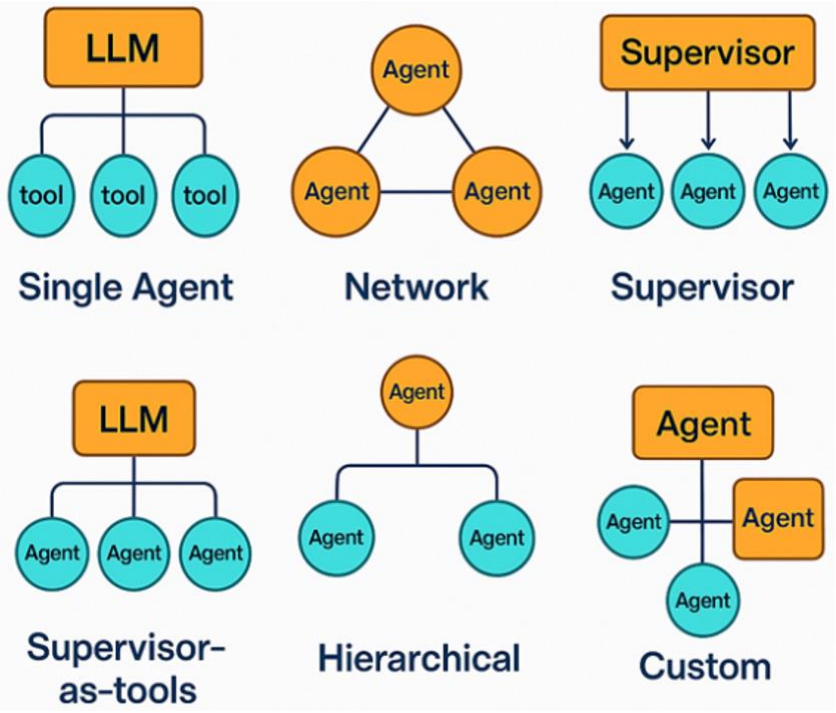

Figure 3.
Common agent architecture.

- Single Agent: Only one central LLM agent integrates all necessary tools. This structure is simple but leads to overload if there are too many tools or diverse tasks.
- Network: Many unstructured, connected agents, each of which can send information to other agents according to its logic [40]. Suitable for systems that require flexible coordination.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

- Supervisor: A Supervisor agent plays a management role, receiving tasks from users and then dividing them among specialized sub-agents before synthesizing the results to provide the final response [41]. This structure simplifies coordination and makes it easy to track the flow of tasks because the interaction flow is primarily centered around one focal point.
- Supervisor-as-tools: turns each specialized agent into a separate "tool" that a main LLM can call [12]. This method is similar to the function calling mechanism but at a higher level: each "tool" is essentially a complex agent with its logic. The benefit is to take advantage of the available LLM pipelines.
- Hierarchical: Agents are organized in a tree structure; the parent agent assigns tasks to child agents, and children can create "grandchild" agents, etc [42]. Suitable for problems with a clear decomposition structure. The advantage is that it is straightforward to control; the disadvantage is that it lacks flexibility if the task changes unexpectedly.
- Custom: a mixed architecture designed according to the specific requirements of the system, which can combine elements of the above models [43]. Applicable to large-scale AI products that need to optimize performance, cost, and reliability, such as a business assistant that integrates multiple components, each being a separate agent.

Among the above structures, the Supervisor model is gaining attention because of its efficiency and ease of implementation. The main idea is to have a central agent that acts as a "coordinator," while the sub-agents are "experts" in different fields [44]. This architecture partly reflects the organizational model in real society: the supervisor agent is like a manager, assigning tasks to employees' sub-agents. Each "employee" specializes in one area, such as searching for information, analyzing data, or programming. The benefit is to take advantage of specialization, and the supervisor will take on the responsibility of deciding on the overall strategy. In the Visual Agentic AI approach proposed by Pati [45], the authors combined the multi-agent supervisor architecture with image processing capabilities, thereby creating a multimodal agent system that can think and see [46]. Specifically, their system consists of a Supervisor Agent as the "brain," coordinating two sub-agents: a Research Agent specializing in searching and synthesizing textual information and a Vision Agent specializing in processing visual requests [47]. When receiving a complex request from a user, the Supervisor will analyze it. If textual information is needed, it will be assigned to the Research Agent for processing. If image analysis is required, the Vision Agent will be contacted. The results from the sub-agents are sent back to the Supervisor to synthesize into the final response. Figure 5 illustrates this architecture: The Supervisor receives the user request, assigns the Research Agent and Vision Agent to perform in parallel, and then merges the results to give the final response [48]. This enables the system to respond to complex queries that require both linguistic knowledge and image analysis, a task that a single agent would struggle to accomplish effectively.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

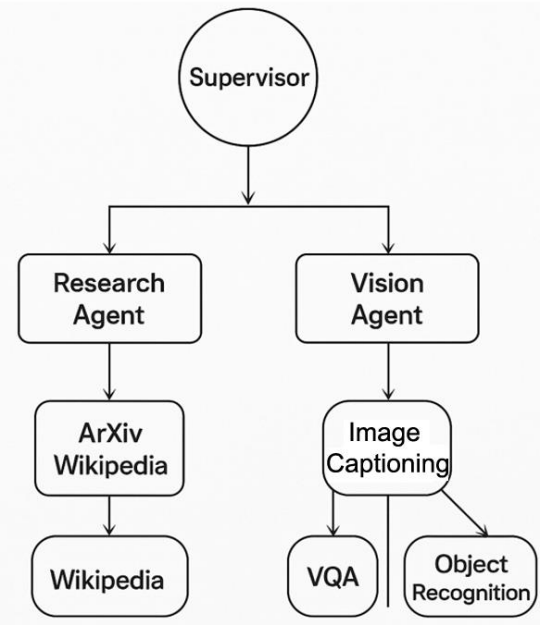

Figure 4.
Multi-model agent architecture with Supervisor.

The multi-agent architecture described above offers many advantages: reducing the load on each model, enabling specialization (for example, the Vision Agent can use a separate LVLM or vision model optimized for image tasks), and facilitating system extension by adding new agents without retraining the entire model. The proof of this is in the Visual Agentic AI system, where additional capabilities can be integrated by attaching an Audio Agent with audio processing tools. The Supervisor then coordinates similarly to the Vision and Research Agents. However, the challenge of multi-agent systems lies in coordinating agents smoothly and communicating effectively. Poor design can lead to issues such as deadlock, inconsistent information sharing, or increased computational costs due to many nested components [49]. To support this, frameworks like LangGraph provide a multi-agent supervisor architectural pattern, allowing programmers to model complex task flows as stateful graphs and easily coordinate specialized agents [50]. For example, LangGraph has a create_supervisor function that creates a supervisor agent node, which interacts with multiple child agent nodes within the same logical graph.

More generally, multi-agent systems are at the cutting edge of research because they open up the possibility of building large, complex AI systems that resemble social systems consisting of many cooperating or competing AI individuals [41]. In the context of AGI, many experts believe that rather than a single giant model, AGI could emerge from a combination of many specialized intelligent agents that can assign tasks, share knowledge, and learn from each other. Examples such as the AgentVerse platform or, more recently, AgentArena, suggest a "universe of agents" that coexist and interact. The multi-agent supervisor architecture is just one of many possible structures. However, it shows that we can build modular, flexible, and extensible AI systems, a practical approach on the path to artificial intelligence that is more complex than single-agent intelligence.

2.6. AI Scientist: Autonomous Scientific Research Agents

One of the most ambitious goals of Agentic AI is to create agents that can perform complex tasks at the expert level, such as independent scientific research. Lu et al. [51] work "The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery" marks the first step toward realizing this goal [52]. They introduce a comprehensive AI agent framework, dubbed AI Scientist, capable of autonomously

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

performing most steps in the open scientific research process, from proposing ideas to publishing results [51]. Specifically, the AI Scientist system enables the integration of multiple AI modules to assume different roles in research. The AI Scientist workflow is designed to mimic the human research process [53]: (1) gather background knowledge from the literature, (2) propose new hypotheses or research ideas, (3) design experimental methods, (4) execute experiments (e.g., run simulations or train models) to collect data, (5) analyze and visualize results, (6) interpret findings and write scientific reports, and (7) perform a peer-review process. An AI Scientist uses advanced LLMs as their "brain," combined with code generation capabilities and access to external tools to perform experiments and analysis.

Lu et al. [51] say their system can generate new research ideas, write code to perform experiments, graph the results, write a complete scientific paper, and then act as a peer reviewer to evaluate the paper itself. All of these steps are done with little to no human intervention. The process can be iterative, and when the evaluation results suggest improved ideas, the system will automatically adjust the hypothesis or experiment and continue the research cycle [54]. In principle, the new knowledge gained after each iteration will be stored in a standard knowledge base to serve as a foundation for subsequent research cycles, simulating the way the human scientific community accumulates knowledge through successive works [55].

The AI Scientist framework has been tested in several areas of AI, including diffusion models, language models, and learning dynamics, with impressive results. With a computational cost of only about $15 per experiment, the system generated drafts of papers containing novel ideas and competing results [53]. To assess quality, the authors also developed an automated reviewer module to critique papers written by AI Scientists. The results showed that this AI reviewer achieved nearly human-level performance in grading papers, and more importantly, papers produced by AI Scientists exceeded the acceptance threshold of a leading conference on machine learning when evaluated by automated reviewers [53]. This suggests that the system can make meaningful scientific contributions, at least according to the initial evaluation criteria.

While it is still early to say that AI scientists can replace scientists, this research opens a promising new direction. On the one hand, it democratizes research as the cost of each intellectual experiment is significantly reduced. Imagine a day when any individual can "deploy" an AI agent to explore science on their behalf, accelerating the pace of knowledge creation [56]. On the other hand, AI scientist demonstrates the potential to combine all the capabilities of agentic AI, including LLM for reasoning, multi-agent systems for role assignment, and tool calling for running code and collecting data. This is a typical example of a complex agent system on the path to AGI, where AI is not only an assistant to the scientist but also gradually becomes a subject participating in the process of scientific creation. In the context of Vietnam, although the application of AI science is still in its early stages, its components, such as experimental code generation support, scientific writing assistants, and automatic review systems, can be integrated into research and training activities. For example, an AI assistant for researchers can suggest reference materials, design simulation experiments, or preliminarily check for logical errors in manuscripts, freeing people from tedious tasks to focus more on core ideas. This will be an exciting direction for the country to increase scientific productivity.

2.7. AgentRxiv: Collaborative AI Agent Research Platform

Suppose an AI Scientist is the vision of a single agent conducting research. In that case, AgentRxiv takes the idea a step further: multiple AI agents collaborate on research together, just as human scientists do in communities [57]. According to Schmidgall and Moor [57], scientific progress rarely comes from a single "Eureka" moment. However, it is often the result of hundreds of researchers working together and building on each other's work. Current agent processes often operate in isolation, with each agent conducting its research without sharing information with other agents [3]. This wastes the opportunity for "mutual learning" between agents. To address this issue, Schmidgall & Moor developed AgentRxiv – a framework that establishes a shared preprint server for agent labs to upload and retrieve each other's research papers, thereby facilitating collaboration, sharing of findings, and

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

incremental building upon the results of AI peers [57]. The name "AgentRxiv" is inspired by arXiv – the open preprint repository for the scientific community. In this system, each "agent lab" can be understood as an agent or a group of agents working together according to a specific research strategy. When an agent completes a result, it posts the report to AgentRxiv, allowing other agents to read and leverage the information [58]. At the same time, the agent can also access the AgentRxiv repository to learn from the work of other agents, thereby avoiding repetition and instead focusing on building upon previous achievements [53].

Initial experiments indicate that collaborating via AgentRxiv enables agents to improve more rapidly than working alone. Specifically, the authors found that agents with access to their previous research results performed significantly better than agents starting from scratch each time [57]. Furthermore, the best strategy they found generalized to other domains, with an average 3.3% improvement in performance [59]. When multiple agent labs share research via AgentRxiv, they can collectively improve faster and achieve higher accuracy [60]. Figuratively speaking, if each agent is an "AI researcher" working in one place, then AgentRxiv is like an "online conference" where they publish and update results, thereby allowing the entire community to move forward faster than any individual. The illustration in the paper envisions a globally distributed network of AI labs connected via AgentRxiv, pursuing a common research goal, say, improving the performance of Math. Humans can provide initial guidance, and then autonomous AI agents can conduct research and publish reports to AgentRxiv, allowing other "AI peers" to access the new findings and adjust their strategies immediately. The result is a group of agents that converge faster and collectively achieve better results than any individual agent could.

AgentRxiv suggests an interesting direction for Agentic AI: building a community of AI agents like a human community. This can be seen as a form of decentralized, collaborative multi-agent system, differing from the supervisor architecture in that no single agent commands all; instead, each agent contributes to the common knowledge. In the future, we can envision AI agents in various settings contributing to an open AgentRxiv for a specific field, for example, "MathAgentRxiv" for mathematics and "ChemAgentRxiv" for chemistry. This not only helps AI agents improve their performance but also serves as a powerful tool for human scientists, as they can utilize AgentRxiv to explore ideas proposed by AI or assign AI tasks to solve time-consuming tasks and synthesize the results.

3. Discussion

Advantages, limitations, and convergence of approaches: Through the above overview, each approach in Agentic AI aims to solve a specific aspect of the problem of building intelligent agents. The table below summarizes some of the main features, advantages, and limitations of each approach.

- LVLMs: The outstanding advantage is the ability to understand and process visual and textual information simultaneously, enabling AI agents to adopt a multimodal view like humans [27]. This lays the foundation for AGI because AGI needs to interact with a diverse real world (vision, sound, language...). The disadvantages of LVLMs are that they require substantial resources, are complex to train, and can degrade performance when encountering data outside their distribution [6]. Fine-tuning for each specific task is also expensive and complicated.
- ReAct agent: The main advantage is its flexibility and high adaptability; the agent can handle unexpected situations, adjusting strategies based on newly acquired information [61]. React is simple to implement and suitable for interactive tasks. The limitation is that it may lack an overall vision, easily fall into loops if not controlled, and may be ineffective for very complex problems that require pre-planning. - Plan-and-Execute agent: The strength lies in its clear structure, which has a global plan, and, therefore, often solves complex problems well with a precise step sequence [62]. It is also easy to analyze and test because the plan is presented in advance. However, the disadvantage is that it lacks flexibility, and a rigid plan can fail if a new, unexpected situation arises. Implementing plan execution is also more complex because it requires separating

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

the two phases and ensuring that the model knows when to switch from planning to execution [18].

- Smolagents: The advantage is that it is lightweight and straightforward, allowing developers to easily customize and understand the agent's operations [15]. The code-first approach effectively leverages the code generation capabilities of LLMs and can reduce ambiguity compared to natural language output. The disadvantage is that it heavily relies on the model's reliability; when generating code, incorrect code can lead to errors or unintended consequences. Additionally, smolagents currently mainly support Python, so it is limited if the task requires a different specific environment.
- Tool Calling: The obvious advantage is that it extends the capabilities of LLM almost indefinitely; it can access new information, make precise calculations, interact with the real world, etc. [63]. This is the key component that turns the model into a real agent. The drawbacks include increased system complexity and security issues. It is necessary to design a reasonable sandbox and limit each tool.
- Visual Agentic: The main advantage is easy specialization and extension - it can combine many agents who are good at different areas, reducing the load on each agent and improving overall efficiency [64]. Supervisor architecture ensures orderly coordination and precise results. The drawbacks include the complexity of inter-agent communication and the need for compelling coordination logic to prevent conflicts or bottlenecks. Moreover, having many agents means that the computational cost can increase.
- AI Scientist: A significant strength is the ability to automate almost the entire scientific research process, integrating many skills (reading, reasoning, planning, writing, experimenting) into a single system [65]. This directly serves the goal of AGI. The disadvantage is that the system is highly complex, requires coordination of many AI components, and cannot operate completely independently. Additionally, AI science is currently applied only in a narrow field (ML research) and assumes a simulation environment; further research is needed to generalize it to other fields.
- AgentRxiv: The advantage lies in creating a mechanism for cooperation and knowledge sharing among agents, thereby enhancing performance and accelerating progress [66]. This represents a novel approach, transitioning agent development from a standalone to a collaborative environment. Limitations may include the risk of sharing incorrect or unverified information (if one agent learns from another agent's incorrect report), like the problem of unverified information on arXiv. A mechanism for assessing the quality of reports is needed to ensure the reliability of shared knowledge.

In general, the above approaches are more complementary than competitive. Many Agentic AI systems today have begun to combine multiple techniques simultaneously to leverage each other's strengths and compensate for their weaknesses. For example, a complete AI assistant may utilize an LLM integrated with tool calling, internally running on the ReAct architecture for flexible inference but occasionally inserting a Plan-and-Execute phase when encountering large tasks that require planning. This agent may include an LVLM as a component to understand both images and text from the user. If the task is too complex, it can activate multi-agent mode, create specialized sub-agents, and act as their supervisor. During operation, these agents can record reports in a shared system for future runs, allowing them to gain experience or for other agents to learn from them. No single solution will meet all AGI's requirements, but clever combinations will get us closer to that goal.

3.1. Convergence Scheme Towards AGI

Based on the above analysis, we propose a convergence architecture scheme that integrates Agentic AI approaches, aiming to build a general intelligent agent in the future.

- At the foundational level, the system will have one or several core LLMs/LVLMs, serving as the central brain responsible for language understanding and background knowledge. LVLM plays the role

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

of processing multi-modal knowledge to help the agent "feel" the visual world, while traditional LLMs ensure language skills and logical reasoning.

- The system operates according to the extended ReAct mechanism, meaning that LLMs will interact in a loop with the environment, including users and external tools. At each step, LLMs can decide to Plan or Act immediately. This decision is based on the complexity of the problem; if the LLM finds the task complex, it switches to Plan-and-Execute mode to ensure a general strategy; if the task is simple or the situation is uncertain, it uses the ReAct reflex mode to respond flexibly.
- The system integrates a rich set of tools and has an automatic tool-calling mechanism. When the central LLM encounters a request that exceeds its capacity, it generates a request to call the corresponding tool. These tools include knowledge query APIs, databases, scientific calculations, and specialized modules such as image, video, and audio processing. The process of calling tools and receiving results is fully automated and closed in the agent's inference loop.
- Regarding the multi-agent architecture, the system will adhere to a multi-agent hierarchical model. At the top layer is the "Master" agent, which is responsible for analyzing the overall task and breaking it down into sub-tasks. The master agent will then create or activate specialized sub-agents to solve each sub-task. These sub-agents are essentially small LLMs or AI modules designed and trained for a specific task. The master agent will play a coordinating role, arranging the order of tasks, collecting results from sub-agents, and checking and synthesizing them into the result.
- Each agent operates according to the principle of the lower layer agent: that is, it can also utilize ReAct and call tools. For example, the "image analysis" sub-agent can internally call a vision model to recognize objects and then write its report. The "coding" child agent can utilize small agents, which means it contains an LLM loop that generates and executes code. This nested design enables the overall system to be multi-layered, versatile, and highly flexible in mobilizing the right capabilities needed for each task.
- All intermediate results and new knowledge generated during the agent's work will be pushed into a common knowledge repository of the system. This repository stores the implemented plans, tools, and results, along with the conclusions drawn. For each new task, the host agent can query this repository to determine if a similar task has been solved in the past, thereby leveraging available results instead of starting from scratch. If the old results are not satisfactory, the system will recognize the need to avoid that approach and try a new solution.
- In principle, this unified Agentic AI system can be deployed at the scale of a large agent running on a single machine or many small agents communicating over a network. The multi-small agent model is suitable when resources are distributed or when you want to leverage the "power of the crowd" of agents. In such an environment, a standard protocol is necessary for agents to exchange reports and requests with each other coherently.
- Finally, to move toward AGI, the system must be able to learn: that is, after each completed task, it does not forget but continuously updates the model parameters so that it can perform better next time. For example, if a child agent writes code that contains an error, the agent will avoid making the same mistake next time. If the governing agent finds that task allocation type A is more efficient than type B over several trials, it will gradually favor type A. This feedback and adjustment mechanism can be implemented through reinforcement learning or by simply having a rule evaluation and editing module.

The above scheme, therefore, combines the most quintessential components: a powerful LLM/LVLM brain, multi-step inference skills, offline capabilities, team organization, deep automation, and knowledge collaboration. This is the shape of a great AI agent that we hope to develop to undertake complex, long-term goals in diverse real-world environments, with performance far exceeding that of individual modules. Of course, this is only a vision at the moment, and there are many challenges to be

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

addressed, including ensuring reliability, efficiency, and, exceptionally, the ability of humans to control such a complex and intelligent system. However, the first pieces have gradually appeared, and each research on Agentic AI is a part of the overall picture. The task of the AI community is to assemble them correctly and fill in the missing pieces on the road to conquering AGI.

3.2. Potential Applications of Agentic AI in Vietnam

Agentic AI technology offers numerous application opportunities in education, research, and industry, especially when tailored to Vietnam's specific context and needs. Below, we review some promising application directions.

In education, AI agents can act as intelligent teaching assistants or virtual tutors to support personalized learning. For example, a system combining LLM and LVLM can help students solve math problems by providing step-by-step instructions, accompanied by illustrations or videos as needed. This agent knows how to use the CAS tool to check results, call the Wikipedia API to look up additional information, and self-adjust explanations to suit the learner's level. In Vietnam, where the demand for online learning and self-study is increasing, an "AI tutor" fluent in Vietnamese and understanding the Vietnamese education program will be a valuable tool to improve the quality of learning. Moreover, AI agents can operate 24/7 and are cost-effective, allowing them to bring knowledge to areas lacking qualified teachers.

Another application is to support teachers in preparing lesson plans and digital content. AI agents can search for reference materials, draft lecture outlines, create multiple-choice questions, and even generate lecture slides. These time-consuming tasks can be automated, allowing teachers to focus more on their creative expertise and pedagogical interactions. In particular, with the help of AgentRxiv or sharing communities, an agent can update the latest scientific materials (for example, modern STEM teaching methods from international conferences) and suggest them for Vietnamese teachers to apply. In scientific research, systems such as AI Scientist and AgentRxiv can effectively support researchers. In Vietnam, where research resources are limited, leveraging AI agents can create a significant leap in productivity. For example, a small research team can utilize an AI agent to scan thousands of papers on arXiv each month, summarize relevant results, and suggest new ideas, tasks that are challenging for humans to accomplish in the era of "information explosion." Domestic AI labs can also connect their agents to global knowledge-sharing platforms, enabling them to participate in the international scientific community at unprecedented speed.

In the field of basic science, AI agents can help simulate chemical experiments and analyze biological data, as well as aid in physics. Instead of having to test many cases manually, researchers can assign AI agents to set up and run mass simulations and then synthesize the outstanding results. For example, in pharmacology, an AI agent could automatically try to "synthesize" chemical compounds and screen which ones have the potential to become drugs, saving lab effort and narrowing the search space for scientists.

In the technology and industry sectors, Agentic AI promises to create highly flexible virtual assistants for businesses. A close example is the intelligent office assistant: an AI agent can manage emails, schedules, and reminders, and automatically schedule meetings. If integrated with a multi-agent system, it can handle multiple tasks: one agent is responsible for scheduling, another agent reads and summarizes important emails every morning, and another agent monitors market news related to the business. With tool-calling capabilities, this assistant can connect to APIs of many services to execute work seamlessly. In manufacturing and operations, AI agents can act as assistants to factory operations, monitoring data from IoT devices, detecting abnormalities, and proactively suggesting adjustments. For example, a power plant could have an AI agent monitor the electricity flow. If it detects a spike in consumption, it could automatically investigate the cause and then recommend increasing capacity or issuing maintenance alerts. Decisions that previously required engineers to make can now be assisted or partially implemented by an AI agent.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

In the field of customer service, Agentic AI agents are superior to conventional chatbots in that they can proactively solve problems for customers rather than respond. For example, when a customer inquires about the status of an order, the agent not only responds but also proactively checks the system and sends a request email if a delay is detected. Does the customer need technical support? The agent can guide users through various operations and even control some, if authorized, all automatically and quickly.

For the government and public services, Agentic AI can help build electronic citizen assistance. Citizens can ask an AI assistant about administrative procedures. The assistant will play the role of a "virtual civil servant" to guide them through filling out the application, checking for errors, and looking up the application's progress through the public service portal, thereby reducing the burden on the administrative apparatus. In Vietnam, where the government is pushing for digital transformation, AI agents well-versed in local laws and procedures will undoubtedly be a key component of future service portals.

Implementation notes: Despite its great potential, the application of Agentic AI also presents specific challenges in Vietnam. First of all, the issue of language and data: Most current models and systems are developed in English, so it is necessary to make efforts to develop Vietnamese models and integrate local knowledge so that AI agents can operate in the proper context. Second, it is necessary to focus on evaluating and testing the system before applying it to sensitive areas; this requires coordination between technical experts and industry experts to ensure that AI gives the right advice, avoiding mistakes that cause consequences. Third, the issue of ethics and privacy: AI agents can act, so the risk of abuse is also higher. There needs to be a legal framework to manage this, for example, by regulating what AI is allowed to do automatically and what requires human consent. Vietnam can learn from international standards on AI while also developing its own standards that are consistent with Vietnamese cultural values and laws.

4. Conclusion

The rise of Agentic AI marks a significant shift from passive AI models to active, autonomous AI systems capable of multidimensional interactions with their environments. This paper has presented an overview of the main approaches shaping the new generation of AI agents – from multimodal LVLM, reasoning agent architectures such as ReAct and Plan-and-Execute, "small but powerful" smolagents libraries, tool calling techniques that extend LLM intelligence, to complex multi-agent systems and autonomous research agents such as AI Scientist and AgentRxiv. Each approach contributes a piece to the overall puzzle toward AGI: models provide knowledge and representation, methodologies organize thinking and action, and systems coordinate and learn over time.

- The takeaway is that no single solution is enough to achieve AGI, but rather intelligent combinations of solutions will get us closer to the goal. Converging trends have already begun to emerge: new LLM models (like GPT-4) already support function calling; platforms like HuggingFace Transformers also integrate tool use and memory modules to support agent construction; many open-source projects are experimenting with combining multi-agent with LLM, etc. Soon, we can expect unified frameworks where programmers can "build mini-AGI" by assembling components, such as language models, toolkits, child agents, and long-term memory mechanisms, without having to reinvent the wheel. For AI scientists and developers in Vietnam to catch up with the Agentic AI trend, we would like to propose some recommendations as follows:
- Research and develop a large Vietnamese language model: Initial steps have been taken, but it is necessary to continue expanding the scale and quality. The domestic model will help deploy agents to better understand Vietnamese people. At the same time, it is necessary to build Vietnamese multi-modal datasets to train and evaluate LVLM in Vietnam.
- Build an open-source Agentic AI library: Take advantage of international frameworks and develop more modules suitable for the Vietnamese environment. This library should be contributed to by

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

the community, both to accelerate domestic R&D and to establish connections with the international community through sharing tools and agents.

- Human resource training: Introduce content about Agentic AI in undergraduate and postgraduate training programs in IT and artificial intelligence. Encourage students to participate in open-source projects on agents, consider offering open short-term courses or conferences to update their knowledge for those working in the industry. A broad understanding will enable Vietnam to apply this technology effectively.
- Pilot application in key areas: Select a few specific problems in Vietnam with high feasibility for agent application. For example, agents supporting farmers and agents supporting doctors in image diagnosis. Small pilot projects will demonstrate practical effectiveness and lessons learned before being replicated.
- Focus on ethical and legal factors: From the R&D stage, a team of experts in AI ethics and law should be involved to develop guidelines for safe agent use and propose policy adjustments as necessary. For example, regulations on responsibility when AI agents cause errors, the privacy of user data in long-term memory agent systems, and human supervision mechanisms when agents operate in sensitive areas.

Looking to the future, Agentic AI will likely be one of the main pillars of next-generation AI. If LLM is the brain, then Agentic AI is how that brain can have a "body" and "behavior" in the real world. Vietnam, with a large and determined young intellectual force, has all the conditions to welcome and contribute to this wave. Proactive research, learning, and early application will help us narrow the gap with leading countries while leveraging AI to address numerous national challenges. Hopefully, the overview in this article will be the first step in providing basic knowledge for Vietnamese scientists and educators, thereby promoting "Made in Vietnam" Agentic AI initiatives in the coming time.

Transparency:

The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.

Acknowledgements:

This research is funded by the Science and Technology Development Fund under grant number B2024-DN03-11.

References

[1] A. Plaat, M. van Duijn, N. van Stein, M. Preuss, P. van der Putten, and K. J. Batenburg, "Agentic large language models, a survey," *arXiv preprint arXiv:2503.23037*, 2025. http://arxiv.org/abs/2503.23037

[2] V. S. V. Jandhyala, "GPT-4 and beyond: Advancements in AI language models," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 10, no. 5, pp. 274–285, 2024. https://doi.org/10.32628/CSEIT241051019

[3] R. Sapkota, K. I. Roumeliotis, and M. Karkee, "Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges," *arXiv preprint arXiv:2505.10468*, 2025. https://doi.org/10.48550/arXiv.2505.10468

[4] L. Wang *et al.*, "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, p. 186345, 2024. https://doi.org/10.1007/s11704-024-40231-1

[5] R. Raman, R. Kowalski, K. Achuthan, A. Iyer, and P. Nedungadi, "Navigating artificial general intelligence development: Societal, technological, ethical, and brain-inspired pathways," *Scientific Reports*, vol. 15, no. 1, pp. 1-22, 2025. https://doi.org/10.1038/s41598-025-92190-7

[6] P. Xu *et al.*, "Lvlm-ehub: A comprehensive evaluation benchmark for large vision-language models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 3, pp. 1877 - 1893, 2024. https://doi.org/10.1109/TPAMI.2024.3507000

[7] A. Munde, "Evaluation of Vision Language Model on UML Diagrams," 2025.

[8] E. Vendrow *et al.*, "INQUIRE: A natural world text-to-image retrieval benchmark," *Advances in Neural Information Processing Systems*, vol. 37, pp. 126500-126514, 2024. https://doi.org/10.52202/079017-4018

[9] D. Sayers *et al.*, "The Dawn of the Human-Machine Era: A forecast of new and emerging language technologies," 2021. https://doi.org/10.17011/jyx/reports/20210518/1

[10] S. Mohammed *et al.*, "The effects of data quality on machine learning performance on tabular data," *Information Systems*, vol. 132, p. 102549, 2025. https://doi.org/10.1016/j.is.2025.102549

[11] V. B. Parthasarathy, A. Zafar, A. Khan, and A. Shahid, "The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities," *arXiv preprint arXiv:2408.13296*, 2024. https://doi.org/10.48550/arXiv.2408.13296

[12] J. Luo *et al.*, "Large language model agent: A survey on methodology, applications and challenges," *arXiv preprint arXiv:2503.21460*, 2025. https://doi.org/10.48550/arXiv.2503.21460

[13] Y. Liu *et al.*, "How ai processing delays foster creativity: Exploring research question co-creation with an llm-based agent," in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024.

[14] S. Yao *et al.*, "React: Synergizing reasoning and acting in language models," *arXiv:2210.03629*, 2022. https://doi.org/10.48550/arXiv.2210.03629

[15] Y. Zhang, Y. Yang, J. Shu, X. Wen, and J. Sang, "Agent models: Internalizing chain-of-action generation into reasoning models," *arXiv preprint arXiv:2503.06580*, 2025. https://doi.org/10.48550/arXiv.2503.06580

[16] Q. Wu *et al.*, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," *arXiv:2308.08155*, 2023. https://doi.org/10.48550/arXiv.2308.08155

[17] U. Turan, Y. Fidan, and C. Yıldıran, "Critical thinking as a qualified decision-making tool," *Journal of History Culture and Art Research*, vol. 8, no. 4, pp. 1–18, 2019. https://doi.org/10.7596/taksad.v8i4.2316

[18] G. He, G. Demartini, and U. Gadiraju, "Plan-then-execute: An empirical study of user trust and team performance when using llm agents as a daily assistant," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025.

[19] H. Taherdoost and M. Madanchian, "Decision making: Models, processes, techniques," *Cloud Computing and Data Science*, pp. 1-14, 2024. https://doi.org/10.37256/ccds.5120243284

[20] D. Horne, "The agentic ai mindset–a practitioner's guide to architectures, patterns, and future directions for autonomy and automation," in *International Conference on the AI Revolution (pp. 434-455). Cham: Springer Nature Switzerland*, 2025.

[21] J. Wang and Z. Duan, "Intelligent spark agents: A modular langgraph framework for scalable, visualized, and enhanced big data machine learning workflows," *arXiv preprint arXiv:2412.01490*, 2024. https://doi.org/10.48550/arXiv.2412.01490

[22] K. Y. Kai, "Adaptability and flexibility in architecture concepts & theories applied in residential architecture to achieve adaptability," *Taylors University Malaysia, AMC Architects Sdn. Bhd, Malaysia*, 2022.

[23] F. Huseynov, *Chatbots in digital marketing: Enhanced customer experience and reduced customer service costs. In Contemporary approaches of digital marketing and the role of machine intelligence*. Hershey, PA, USA: IGI Global, 2023.

[24] A. Li, Y. Xie, S. Li, F. Tsung, B. Ding, and Y. Li, "Agent-oriented planning in multi-agent systems," *arXiv preprint arXiv:2410.02189*, 2024. https://doi.org/10.48550/arXiv.2410.02189

[25] H. Chen and Y. Ding, "Implementing traffic agent based on LangGraph," in *Fourth International Conference on Intelligent Traffic Systems and Smart City (ITSSC 2024) (Vol. 13422, pp. 582-587). SPIE*, 2025.

[26] X. Liu *et al.*, "Prompting frameworks for large language models: A survey," *arXiv preprint arXiv:2311.12785*, 2023. https://doi.org/10.48550/arXiv.2311.12785

[27] X. Wang, B. Zhuang, and Q. Wu, "Are large vision language models good game players?," *arXiv preprint arXiv:2503.02358*, 2025. https://doi.org/10.48550/arXiv.2503.02358

[28] K. Feng *et al.*, "Cocoa: Co-planning and co-execution with ai agents," *arXiv preprint arXiv:2412.10999*, 2024. https://doi.org/10.48550/arXiv.2412.10999

[29] M. Mitchell, A. Ghosh, A. S. Luccioni, and G. Pistilli, "Fully autonomous ai agents should not be developed," *arXiv preprint arXiv:2502.02649*, 2025. https://doi.org/10.48550/arXiv.2502.02649

[30] X. Ye, M. Zhang, and S. Wu, "Open Problems and a Hypothetical Path Forward in LLM Knowledge Paradigms," *arXiv preprint arXiv:2504.06823*, 2025. https://doi.org/10.48550/arXiv.2504.06823

[31] W. Liu *et al.*, "Toolace: Winning the points of llm function calling," *arXiv preprint arXiv:2409.00920*, 2024. https://doi.org/10.48550/arXiv.2409.00920

[32] I. Gim, S.-s. Lee, and L. Zhong, "Asynchronous LLM Function Calling," *arXiv preprint arXiv:2412.07017*, 2024. https://doi.org/10.48550/arXiv.2412.07017

[33] Y. Zhuang, Y. Yu, K. Wang, H. Sun, and C. Zhang, "ToolQA: A dataset for LLM question answering with external tools," *arXiv:2306.13304*, 2023. https://doi.org/10.48550/arXiv.2306.13304

[34] P. P. Ray, "A review on Agent-to-Agent protocol: Concept, state-of-the-art, challenges and future directions (TechRxiv Preprint)," *TechRxiv*, 2025. https://doi.org/10.36227/techrxiv.174612014.42157096/v1

[35] C. A. Wrenn, "Can autonomous technology reduce the driver shortage in the commercial trucking industry," Doctoral Dissertation, California Southern University, 2017.

[36] R. Xu and G. Li, "A comparative study of offline models and online llms in fake news detection," *arXiv preprint arXiv:2409.03067*, 2024. https://doi.org/10.48550/arXiv.2409.03067

[37] T. G. Pelser and G. Gaffley, "Implications of digital transformation on the strategy development process for business leaders. In Promoting inclusive growth in the fourth industrial revolution." Hershey, PA, USA: IGI Global, 2020.

[38] S.-L. Wamba-Taguimdje, S. Fosso Wamba, J. R. Kala Kamdjoug, and C. E. Tchatchouang Wanko, "Influence of artificial intelligence (AI) on firm performance: The business value of AI-based transformation projects," *Business process management journal*, vol. 26, no. 7, pp. 1893-1924, 2020. https://doi.org/10.1108/BPMJ-10-2019-0411

[39] Y. Yang, Q. Peng, J. Wang, Y. Wen, and W. Zhang, "LLM-based multi-agent systems: Techniques and business perspectives," *arXiv preprint arXiv:2411.14033*, 2024. https://doi.org/10.48550/arXiv.2411.14033

[40] S. M. Thampi, "Survey of search and replication schemes in unstructured p2p networks," *arXiv preprint arXiv:1008.1629*, 2010. https://doi.org/10.48550/arXiv.1008.1629

[41] D. Maldonado, E. Cruz, J. A. Torres, P. J. Cruz, and S. d. P. G. Benitez, "Multi-agent systems: A survey about its components, framework and workflow," *IEEE Access*, vol. 12, pp. 80950-80975, 2024. https://doi.org/10.1109/ACCESS.2024.3409051

[42] M. Lauria, A. Chakravarti, and G. Baumgartner, *Self-organizing scheduling on the Organic Grid. In M. Parashar & S. Hariri (Eds.), Autonomic computing: Concepts, infrastructure, and applications*. Boca Raton, FL, USA: CRC Press, 2006.

[43] J. Zhang, "The architecture of p2p computer collaborative design system based on artificial intelligence," in *In Journal of Physics: Conference Series (Vol. 1574, No. 1, p. 012090). IOP Publishing*, 2020.

[44] G. Charles, J. Freeman, and T. Garfat, *Supervision in child and youth care practice*. Cape Town, South Africa: CYC-Net Press, 2016.

[45] A. K. Pati, "Agentic AI: A comprehensive survey of technologies, applications, and societal implications," *IEEE Access*, vol. 13, pp. 151824–151837, 2025. https://doi.org/10.1109/ACCESS.2025.3585609

[46] U. M. Borghoff, P. Bottoni, and R. Pareschi, *An organizational theory for multi-agent interactions*. Bridging Human Agents: LLMs, and Specialized AI, 2025.

[47] C. Zhang *et al.*, "Large language model-brained gui agents: A survey," *arXiv preprint arXiv:2411.18279*, 2024. https://doi.org/10.48550/arXiv.2411.18279

[48] B. Manaris, "Natural language processing: A human-computer interaction perspective," *Advances in Computers*, vol. 47, pp. 1-66, 1998.

[49] A. Javaid, *Understanding operating systems: Principles and practices*. Lahore, Pakistan: Nexus Academic Publishers (NAP), 2013.

[50] G. d. A. e Aquino *et al.*, "From RAG to multi-agent systems: A survey of modern approaches in LLM development," 2025.

[51] C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha, "The ai scientist: Towards fully automated open-ended scientific discovery," *arXiv preprint arXiv:2408.06292*, 2024. https://doi.org/10.48550/arXiv.2408.06292

[52] H. Sakib, E. Zachary, and C. M. Aakash Bansal, "Semantic similarity metrics for evaluating source code summarization," presented at the IEEE International Conference on Program Comprehension, Apr. 2022, pp. 36–47, 2022.

[53] Y. Yamada *et al.*, "The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search," *arXiv preprint arXiv:2504.08066*, 2025. https://doi.org/10.48550/arXiv.2504.08066

[54] S. Ivanova, J. Lee, and K. Jane, "Iterative Design: A cyclical process of design, testing, and refinement," 2024.

[55] D. Yang, L. Zeng, J. Rao, and Y. Zhang, "Knowing You don't know: Learning when to continue search in multi-round rag through self-practicing," in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 1305-1315)*, 2025.

[56] A. Przegalińska and D. Jemielniak, *Strategizing AI in Business and Education: Emerging technologies and business strategy*. . Cambridge, United Kingdom: Cambridge University Press, 2023.

[57] S. Schmidgall and M. Moor, "AgentRxiv: Towards collaborative autonomous research," 2025. http://arxiv.org/abs/2503.18102

[58] S. Schulhoff *et al.*, "The prompt report: A systematic survey of prompt engineering techniques," 2025. http://arxiv.org/abs/2406.06608

[59] A. Podolsky, T. Kini, and L. Darling-Hammond, "Does teaching experience increase teacher effectiveness? A review of US research," *Journal of Professional Capital and Community*, vol. 4, no. 4, pp. 286-308, 2019. https://doi.org/10.1108/JPCC-12-2018-0032

[60] S. Schmidgall *et al.*, "Agent laboratory: Using llm agents as research assistants," *arXiv preprint arXiv:2501.04227*, 2025. http://arxiv.org/abs/2501.04227

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983

[61] D. Settembre-Blundo, R. González-Sánchez, S. Medina-Salgado, and F. E. García-Muiña, "Flexibility and resilience in corporate decision making: A new sustainability-based risk management system in uncertain times," *Global Journal of Flexible Systems Management*, vol. 22, no. Suppl 2, pp. 107-132, 2021. https://doi.org/10.1007/s40171-021-00277-7

[62] M. Parmar *et al.*, "PlanGEN: A multi-agent framework for generating planning and reasoning trajectories for complex problem solving," 2025. http://arxiv.org/abs/2502.16111

[63] Y. Qin *et al.*, "ToolLLM: Facilitating large language models to master 16000+ real-world APIs," 2023. http://arxiv.org/abs/2307.16789

[64] A. Ramachandran, "A survey of agentic AI, multi-agent systems, and multimodal frameworks: Architectures, applications, and future directions," Unpublished Manuscript / Preprint. Available at ResearchGate, 2024.

[65] T. Dorigo *et al.*, "Artificial intelligence in science and society: The vision of USERN," *IEEe Access*, 2025. https://doi.org/10.1109/ACCESS.2025.3529357

[66] H. Ma, Z. Luo, T. V. Vo, K. Sima, and T.-Y. Leong, "Centralized reward agent for knowledge sharing and transfer in multi-task reinforcement learning," *arXiv preprint arXiv:2408.10858*, 2024. https://doi.org/10.48550/arXiv.2408.10858

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 10, No. 2: 73-90, 2026
DOI: 10.55214/2576-8484.v10i2.11983