# The Infinitus Framework: A Conceptual Architecture for a Self-Improving, Continually Learning, and Fully Autonomous Multi-Agent System

**Vassil Nikolov, operalytix.com**

**Abstract**

This paper introduces the Infinitus Framework, a conceptual architecture for a new class of autonomous AI. This system is designed for perpetual, 24/7 operation with effectively infinite context and delivery capabilities. At its core, the framework leverages a multi-agent system (MAS) where a central Orchestrator agent performs recursive, self-improving task decomposition, distributing sub-tasks to a network of specialized Executor agents. These Executors are equipped with intelligent tool selection mechanisms and are grounded in a user-provided, offline knowledge base via Retrieval-Augmented Generation (RAG). We detail the key technological pillars required for such a system: advanced memory architectures to achieve infinite context, continual learning paradigms to enable infinite activity, and a robust design for secure, air-gapped deployment. Furthermore, we analyze the critical systemic challenges, including error propagation, computational overhead, security vulnerabilities, and the necessity of formal verification and explainability (XAI). Finally, we explore the transformative applications and profound ethical implications of developing such powerful autonomous systems.

## Introduction

### The Current Frontier of Autonomous AI: From Single Agents to Collaborative Systems

The field of artificial intelligence has undergone a profound transformation, evolving from the development of monolithic, task-specific models to the engineering of sophisticated autonomous agents. The advent of Large Language Models (LLMs) has been a primary catalyst for this shift, demonstrating human-like capabilities in language fluency, reasoning, and even achieving human-level performance in specific domains.[1] This has led to a surge in research focused on LLM-based autonomous agents—systems that can perceive their environment, formulate plans, and execute actions to achieve

goals with minimal human intervention.[1]

However, the limitations of a single agent operating in isolation quickly become apparent when faced with complex, multi-faceted problems. Consequently, the frontier of AI research is rapidly moving towards Multi-Agent Systems (MAS), computerized systems composed of multiple interacting intelligent agents.[4] The fundamental premise of MAS is that a collective of coordinated agents can solve problems that are difficult or impossible for an individual agent or a monolithic system to address.[5] This paradigm represents the next logical step in unlocking productivity gains, moving from single-threaded task execution to parallelized, collaborative problem-solving.[6] The emergence of LLM-based multi-agent systems, where each agent is powered by a capable language model, has opened up a new and fertile area of research, enabling far more sophisticated interactions, negotiations, and coordinated behaviors than ever before.[5] It is within this context—the transition from singular to collective intelligence—that the need for a more advanced, comprehensive architectural framework becomes not only apparent but necessary.

## Limitations of Existing Architectures: Context, Learning, and Deployment Constraints

Despite the promise of MAS, the realization of truly autonomous, perpetually operating systems is hindered by a triad of fundamental, interconnected limitations inherent in current AI architectures. These constraints are not independent issues but a tightly coupled set of challenges where a naive solution for one often exacerbates the others.

First, **context window limitations** represent a significant architectural bottleneck. The self-attention mechanism, the cornerstone of the dominant Transformer architecture, exhibits a computational and memory complexity that scales quadratically with the length of the input sequence, typically denoted as O(n2) where n is the sequence length.[7] This quadratic scaling makes it computationally infeasible to process very long sequences of text, such as entire books, codebases, or extended conversation histories. As a result, developers are forced to implement complex and often inefficient workarounds like chunking data and using separate retrieval systems to feed relevant context into the model's limited window, a process that can lose critical long-range dependencies and coherence.[7]

Second, existing models suffer from **static knowledge and catastrophic forgetting**. AI models are typically trained on a massive but fixed dataset, creating a snapshot of knowledge at a particular point in time. They struggle to incrementally and sequentially acquire new knowledge without severely degrading their performance on previously learned tasks—a phenomenon known as catastrophic forgetting.[2] This inability to learn continuously prevents the development of systems capable of "infinite activity" or true lifelong learning, forcing a costly and disruptive cycle of taking models offline for complete retraining to incorporate new information.

Third, current systems face severe **deployment and security constraints**. The most powerful models are predominantly accessible via cloud-based APIs, which raises significant concerns regarding data privacy,

security, and operational reliability, especially when handling sensitive corporate intellectual property, financial data, or personal information.[10] The reliance on an internet connection makes these systems vulnerable to outages and precludes their use in secure, air-gapped environments common in defense, critical infrastructure, and research sectors where data confidentiality is paramount.[11]

The deep interplay between these limitations is a critical barrier. For instance, an attempt to solve the context problem by simply using a much larger model with a bigger window dramatically increases deployment costs and hardware requirements, making secure offline deployment even more difficult. Similarly, designing an offline system for continual learning is challenging because it cannot rely on the continuous stream of web-scale data that many learning paradigms implicitly assume, making the quality of its adaptation heavily dependent on the history of its own tasks and user-provided knowledge. A holistic framework must therefore seek an equilibrium that addresses these interconnected challenges synergistically rather than in isolation.

## Thesis: Proposing the Infinitus Framework for Infinite-Scale Autonomous Operation

This paper proposes the Infinitus Framework, a conceptual architecture designed as a holistic solution to the aforementioned limitations. The central thesis is that by systematically integrating distinct but complementary fields of research—advanced multi-agent system design, recursive task decomposition, infinite-context memory architectures, continual learning paradigms, and offline Retrieval-Augmented Generation (RAG)—it is possible to formulate a cohesive blueprint for a system that is perpetually active, contextually unbounded, and securely deployable.

The Infinitus Framework is architected around an **Orchestrator-Executor model**. A central Orchestrator agent is responsible for receiving complex, high-level tasks and performing a novel, self-improving recursive decomposition to break them into a hierarchy of manageable sub-tasks. These sub-tasks are then distributed to a dynamic network of specialized Executor agents. To overcome memory constraints, the framework incorporates a heterogeneous memory system, combining compressive and episodic memory architectures to achieve a functionally infinite context. To enable perpetual operation, it embeds continual learning mechanisms, allowing the system to learn from every task it performs without catastrophic forgetting. Finally, to ensure secure and reliable delivery, the entire system is designed for offline, air-gapped deployment, grounding its knowledge and reasoning in a user-provided, private knowledge base via a robust RAG pipeline.

## Paper Structure and Contributions

This paper is structured to provide a comprehensive exploration of the Infinitus Framework, beginning

with its theoretical underpinnings and culminating in a discussion of its practical applications and societal implications. Section 2 reviews the foundational paradigms in autonomous systems, including MAS, task decomposition, and hierarchical planning. Section 3 presents the detailed conceptual architecture of the Infinitus Framework, detailing the roles of the Orchestrator and Executor agents and their communication backbone. Section 4 delves into the core technological capabilities required for infinite-scale operation, namely infinite context, infinite activity, and infinite delivery. Section 5 provides a rigorous analysis of the critical systemic challenges that emerge from such a complex architecture, including error propagation, security, scalability, and the need for verification and explainability. Section 6 explores the transformative applications of the framework and confronts its profound ethical implications. Finally, Section 7 offers a conclusion and outlines directions for future research.

The primary contributions of this paper are threefold:

1. The synthesis of disparate, cutting-edge research areas into a single, cohesive, and novel conceptual architecture for a next-generation autonomous AI system.
2. The identification and analysis of the second- and third-order systemic challenges that emerge from the deep integration of these technologies, moving beyond component-level problems to a holistic system-level analysis.
3. A forward-looking discussion on the transformative applications and the essential ethical guardrails required to guide the responsible development of such powerful autonomous technology.

## Foundational Paradigms in Autonomous Systems

The design of the Infinitus Framework is not conceived in a vacuum; it stands on the shoulders of decades of research in artificial intelligence. Its architecture is a deliberate synthesis of established principles from multi-agent systems, classical and modern planning techniques, and the modular design of contemporary LLM-based agents. Understanding these foundational paradigms is essential to appreciating the framework's structure and rationale.

## The Principles of Multi-Agent Systems (MAS)

A Multi-Agent System (MAS) is formally defined as a computerized system composed of a set of interacting, intelligent elements, known as agents, that work in concert to achieve a common purpose.[4] These systems are particularly adept at solving problems that are too large, complex, or geographically distributed for a single agent to handle effectively. The intelligence of a MAS can manifest through various means, including methodical procedures, functional algorithms, or sophisticated reinforcement learning approaches.[5]

## Core Characteristics

Agents within a MAS are defined by several key characteristics that enable robust and flexible collective behavior [5]:

- **Autonomy:** Agents are at least partially independent and self-aware. They have control over their own actions and internal state.
- **Local Views:** No single agent possesses a complete global view of the entire system. Each agent operates based on its own local information and perceptions, a necessary constraint in complex systems where a global state would be too vast to process or maintain.
- **Decentralization:** There is no single, designated controlling agent. This decentralization of command and control is a hallmark of MAS, preventing single points of failure and enabling emergent, self-organizing behavior.

## Coordination and Communication

The true power of a MAS lies in the ability of its agents to coordinate their actions. Without effective coordination, a collection of agents can devolve into a chaotic system working at cross-purposes, leading to inefficiency and failure.[13] Several paradigms have been developed to facilitate this coordination. One common approach is

**contract-based negotiation**, where a task is broadcast, and interested agents respond with bids (e.g., "I can perform this task at this cost"), leading to a dynamically formed contract.[5] Another powerful paradigm is

**stigmergy**, where agents communicate indirectly by modifying their shared environment. This is analogous to ants leaving pheromone trails; one agent leaves a piece of information (a "pheromone") that influences the subsequent actions of other agents that perceive it.[5]

## Architectural Models

The structure of a MAS can range from simple to highly complex. Some systems employ a **linear workflow**, where agents are connected in a chain, similar to a factory assembly line, with the output of one agent becoming the input for the next.[6] More sophisticated systems utilize

**fully connected mesh networks**, where every agent can potentially interact with every other agent, allowing for highly dynamic and flexible collaboration.[6] To bring order to this complexity, conceptual models like the

**FC-MAS (Framework-Components in Multi-Agent System)** have been proposed. FC-MAS provides a standardized model comprising five abstract layers, each serving a specific purpose and detailing the resources and functions required for the deployment and operation of a MAS in practical engineering applications.[4] This structured approach helps simplify the design and integration of these complex systems.

**A Taxonomy of Task Decomposition Strategies**

Task decomposition is a fundamental problem-solving technique in AI, applying the classic "divide and conquer" strategy to machine learning and planning tasks.[14] The core idea is to break a large, complex problem into a set of smaller, simpler sub-tasks that are easier to solve individually. The solutions to these sub-tasks are then composed to form the solution to the original problem.

## Benefits

The engineering advantages of task decomposition are significant and well-documented.[14] It improves

**modularity**, as different models or components can be used to solve different parts of the problem, enhancing system maintainability. It enhances **functional correctness**, as breaking a problem down can help an LLM, much like a human, to produce a higher-quality output on a more focused task. It can improve **scalability** by limiting the output size required from any single model call. It also improves **testability**, as evaluating simpler sub-tasks is more straightforward than validating the output of a single, complex task.[14] Furthermore, by enabling the use of smaller, more specialized models for specific sub-tasks, this approach can be significantly more cost-effective than relying on a single, large, general-purpose model for everything.[15]

## Drawbacks and Pitfalls

Despite its benefits, task decomposition is not a panacea. The primary trade-off is a significant increase in **system complexity**. Introducing multiple models, orchestrators, and communication channels adds more

components to build, maintain, and debug.[14] This added orchestration overhead can also lead to higher

**overall latency**, even if the individual smaller models are faster; the time spent coordinating and passing information between components can negate the performance gains of the sub-tasks themselves.[15]

Perhaps the most subtle drawback is the risk of **over-engineering**, which can stifle the very creativity and novelty that makes LLMs so powerful.[15] By manually breaking a task into a rigid sequence of sub-tasks, the system may fail to discover the serendipitous connections and novel insights that can emerge from a more holistic, end-to-end approach. The process of crafting intricate prompts for each sub-task can also become convoluted, ironically leading to reduced accuracy or increased hallucinations.[15]

## Decomposition Methods

Task decomposition techniques have evolved from simple prompting strategies to more structured, programmatic approaches. A primary distinction can be made between **iterative decomposition** and **single-step decomposition**.[16]

- **Iterative Decomposition:** In this approach, the system generates a single, simple sub-task, executes it, and then uses the result to inform the generation of the next sub-task. This is a sequential, adaptive process. Techniques like DecomP and Successive Prompting, which use a repetitive questioning approach to gather information, fall into this category.[16] The advantage is flexibility, as the plan can adapt based on intermediate results.
- **Single-Step Decomposition:** This approach uses a single prompt or planning phase to break down the entire task into a full sequence or tree of sub-tasks at the outset. The Plan-and-Solve prompting technique is an example of this method.[16] This is more time-efficient as it requires fewer interactions with the LLM, but the resulting plan is static and cannot adapt to unforeseen outcomes during execution.

These methods represent a spectrum, from the simple in-context reasoning of Chain-of-Thought prompting [16] to the creation of complex, hierarchical sub-task trees.[16]

## Hierarchical Planning: From Hierarchical Task Networks (HTN) to Hierarchical Reinforcement Learning (HRL)

Hierarchical planning is a specialized and powerful form of task decomposition that organizes tasks into a formal hierarchy. This approach mirrors human problem-solving, where we naturally think in terms of high-level goals and the lower-level steps needed to achieve them. This concept has been realized through both classical, knowledge-driven methods and modern, learning-based approaches. A sophisticated

system should not be forced to choose between the predictability of classical methods and the adaptability of modern learning; rather, it should hybridize them. The structure of classical Hierarchical Task Networks (HTN) can provide a robust, reliable scaffold for high-level strategic planning, while the learning capabilities of Hierarchical Reinforcement Learning (HRL) can be used to discover novel and more efficient ways to execute the steps within that scaffold. This fusion combines the safety of human-defined knowledge with the power of machine-driven discovery.

## Hierarchical Task Network (HTN)

Hierarchical Task Network (HTN) planning is a classical AI planning paradigm that solves problems by decomposing them according to a pre-defined library of methods.[17] It is particularly effective in domains where tasks have a natural, built-in hierarchical structure, such as robotics or logistics.[18] The core idea is to escape the exponential explosion of searching through all possible sequences of primitive actions by starting the planning process at a more abstract level.[19]

The key components of an HTN planner are [18]:

- **Tasks:** These are the units of work. They can be **primitive tasks**, which correspond to actions that can be directly executed (e.g., pick_up_object), or **compound (abstract) tasks**, which represent high-level goals that require further decomposition (e.g., make_coffee).
- **Methods:** These define how a compound task can be decomposed into a partially ordered set of sub-tasks (which can themselves be primitive or compound). A single compound task can have multiple methods, representing different ways to achieve the same goal. For example, the task travel_to_park could have a "take bus" method and a "take taxi" method, each with its own preconditions (e.g., having bus fare) and sub-tasks.[20]
- **Operators:** These define the primitive tasks. Each operator specifies the preconditions that must be true for it to be executed and the effects it will have on the state of the world upon execution.[18]

HTN planning is a top-down, knowledge-driven process. It begins with a high-level goal and recursively applies methods from its plan library to decompose compound tasks until only a sequence of executable primitive operators remains.[18] This approach is powerful because it leverages human domain knowledge encoded in the methods, resulting in efficient and human-understandable plans.[20] However, its primary limitation is its rigidity; it can only solve problems for which it has pre-defined methods in its library.

## Hierarchical Reinforcement Learning (HRL)

Hierarchical Reinforcement Learning (HRL) is a machine learning framework that applies the principles of hierarchy to reinforcement learning, enabling agents to solve complex, long-horizon tasks more

efficiently.[21] Much like HTN, HRL decomposes a complex task into simpler sub-tasks, but instead of relying on a human-authored library, it

*learns* the hierarchy and the policies for achieving sub-goals from data.[21]

The typical HRL architecture consists of multiple levels of policies [23]:

- A **high-level policy** (or "meta-policy") operates at a temporally abstract level. It does not select primitive actions but instead selects a sequence of sub-goals or "options."
- A set of **low-level policies** are responsible for achieving the sub-goals set by the high-level policy. Each low-level policy takes primitive actions in the environment to reach its assigned sub-goal.

This hierarchical structure provides several key advantages for learning. First, it makes credit assignment easier. Because the high-level policy operates over multiple environmental steps, rewards are propagated much faster, improving learning speed.[22] Second, it facilitates more meaningful exploration. Instead of exploring randomly at the level of atomic actions (e.g., "move finger joint by 1 degree"), the agent can explore at the level of meaningful sub-goals (e.g., "grasp the object"), which is far more efficient for discovering useful behaviors.[23] The primary challenge of HRL is the difficulty of automatically discovering a useful and effective task hierarchy (the "option discovery problem").[22]

## The Modern LLM-based Agent: A Unified Modular Approach

The building blocks of the Infinitus Framework are its individual agents. The design of these agents, particularly the Executor agents, is based on a unified, modular architecture that has emerged as a standard for constructing capable LLM-based autonomous agents.[1] This architecture deconstructs agent functionality into four distinct but interacting modules, providing a clear separation of concerns and a flexible foundation for building specialized capabilities.

The four core modules are [1]:

1. **Profiling Module:** This module defines the agent's identity, role, and persona. It specifies the agent's capabilities, goals, and even its personality traits (e.g., "You are a meticulous Python programmer specializing in data analysis"). This profile is typically encoded in a system prompt that fundamentally influences the LLM's behavior throughout its operation, guiding its planning and actions.[1]
2. **Memory Module:** This module endows the agent with the ability to store and recall information from past interactions, which is crucial for maintaining context, learning from experience, and acting consistently over time. Memory can be structured in different ways, from a simple **unified memory** that uses the LLM's context window as a form of short-term memory, to a more sophisticated **hybrid memory** that combines a short-term buffer with a long-term external database (e.g., a vector store) for persistent knowledge.[1]
3. **Planning Module:** This module is the agent's cognitive engine, responsible for decomposing tasks

and formulating a sequence of steps to achieve its goals. Plans can be generated in a single shot or iteratively. Crucially, advanced planning modules incorporate a **feedback loop**, allowing the agent to revise its plan based on the outcomes of its actions. This feedback can come from the environment (e.g., an error message), from a human supervisor, or even from the model itself in a process of self-reflection and critique.[1]

4. **Action Module:** This module is the agent's interface with the world. It translates the agent's internal decisions and plans into concrete outputs. An agent's action can take many forms, including generating text, communicating with other agents, or, most powerfully, calling **external tools**. By giving agents access to APIs, databases, code interpreters, and web search, the action module allows them to overcome the inherent limitations of the LLM (e.g., its static knowledge) and interact with live data and external systems.[1] The ReAct (Reason+Act) paradigm, where an agent reasons about which action to take, takes the action, and then observes the outcome to inform its next reason-act cycle, is a common and effective pattern for this module.[27]

## The Infinitus Agent Architecture: A Conceptual Framework

Building upon the foundational paradigms of multi-agent systems, hierarchical planning, and modular agent design, we now present the conceptual architecture of the Infinitus Framework. This framework is not a single model but a "system of systems," designed to orchestrate a vast network of autonomous agents to solve complex problems with unparalleled scale and persistence. Its architecture is a deliberate hybrid, blending centralized coordination with decentralized execution to harness the benefits of both approaches.

## High-Level System Overview: The Orchestrator-Executor Model

At its highest level, the Infinitus Framework is structured as a hybrid centralized-decentralized system, which can be termed the **Orchestrator-Executor model**.[13] This architecture consists of two primary components:

1. A single, central **Orchestrator Agent**, which serves as the cognitive core and master coordinator of the system.
2. A large, distributed, and heterogeneous network of **Executor Agents**, which are the specialized "workers" that perform the granular sub-tasks.

This model is analogous to the structure of a human expert team, where a lead researcher or project manager (the Orchestrator) is responsible for understanding a complex problem, breaking it down into constituent parts, and delegating those parts to a team of specialists (the Executors), each with their own

unique skills and tools.[28] The Orchestrator does not perform the work itself; it manages the

*process* of getting the work done.

Communication within this architecture is fundamentally **asynchronous**. The Orchestrator dispatches tasks to the Executors, and the Executors report their results without requiring a synchronous, blocking connection. This decoupling is essential for scalability and is managed by a robust communication backbone, likely implemented using message queues (e.g., Redis, Amazon SQS), a publish-subscribe system, or an event-driven blackboard architecture where agents communicate by posting information to a shared repository.[29] This allows for massive parallelization, as hundreds or thousands of Executor agents can work on their assigned sub-tasks concurrently.

## The Orchestrator Agent: Master of Recursive Decomposition and Distribution

The Orchestrator Agent is the most critical component of the Infinitus Framework. It embodies the system's strategic intelligence, and its primary function is to transform a single, complex, and often ambiguous user goal into a concrete, executable, and optimized plan composed of many simple, well-defined sub-tasks.

## Self-Improving Recursive Decomposition

The core innovation of the Orchestrator lies in its method of task decomposition. It does not rely on a static, pre-programmed planning logic. Instead, it employs a dynamic, self-improving, and recursive decomposition process inspired by cutting-edge research frameworks like **ADaPT (As-Needed Decomposition and Planning for complex Tasks)** and **LADDER (Learning through Autonomous Difficulty-Driven Example Recursion)**.[31]

The decomposition workflow proceeds as follows:

1. **Initial Assessment and Attempt:** Upon receiving a complex goal from a user, the Orchestrator first assesses its complexity. Following the "as-needed" principle from ADaPT, it may first attempt to solve the task with a single, highly capable Executor agent if the task seems manageable.[32] This avoids unnecessary decomposition for simpler problems.
2. **Recursive Decomposition on Failure:** If the initial attempt fails, or if the task is immediately identified as too complex, the Orchestrator invokes its recursive decomposition planner. Drawing inspiration from LADDER, the Orchestrator prompts its own powerful LLM to generate a set of simpler "variants" or sub-tasks of the main goal.[33]
3. **Creating a Difficulty Gradient:** This process is applied recursively. Each newly generated sub-task

that is still too complex is itself decomposed into even simpler sub-variants. This recursive process continues until the Orchestrator generates a hierarchical tree of tasks, where the leaf nodes are **primitive tasks**—actions that are simple, unambiguous, and can be reliably solved by an Executor agent.[16] This creates a "difficulty gradient," where the solutions to the simplest problems at the bottom of the hierarchy provide the necessary information and context to solve the progressively harder parent tasks above them.[34]

This recursive, self-guided decomposition allows the system to bootstrap its own problem-solving process. It leverages the LLM's own reasoning capabilities to figure out *how* to solve a problem, creating a detailed, structured plan from a high-level intent. This process has a profound secondary effect: every decomposition tree, whether it leads to success or failure, becomes a rich, structured training example. This "execution trace" captures not just an input and an output, but the entire reasoning process—the strategy, the sub-goals, the allocation decisions. By feeding these traces back into a continual learning module, the system can use reinforcement learning to improve the Orchestrator's planning capabilities over time. The act of solving problems generates the perfect data to learn how to solve problems better, creating a powerful, system-wide self-improvement loop.

### Dynamic Agent Profiling and Task Allocation

Once the task decomposition tree is generated, the Orchestrator must execute the plan by distributing the sub-tasks to its network of Executor agents. This is a sophisticated resource allocation and scheduling problem. To manage this, the Orchestrator maintains a dynamic, real-time registry of all available Executor agents. For each agent, it maintains a **profile** that includes [1]:

- **Specialization and Skills:** The agent's designated role (e.g., ResearchAgent, CodingAgent).
- **Tool Access:** The set of external tools and APIs the agent is authorized to use.
- **Performance History:** Metrics on its past success rates, latency, and resource consumption for different types of tasks.
- **Current State:** Its current workload and availability.

Using this registry, the Orchestrator acts as an intelligent "dispatcher." It matches each sub-task in the decomposition tree to the most suitable available agent. This delegation process is highly explicit. Following the principles for effective delegation in MAS, the Orchestrator provides each Executor with a clear and detailed "task description" that includes [28]:

- A precise **objective** for the sub-task.
- A specified **output format** to ensure its results can be integrated back into the main plan.
- Guidance on which **tools and data sources** to use.
- Clear **task boundaries** and constraints (e.g., a maximum number of tool calls or a time limit) to prevent resource over-consumption, scaling the allocated effort to the sub-task's complexity.[28]

## Specialized Executor Agents: A Heterogeneous Network of Experts

The Executor agents form the distributed workforce of the Infinitus Framework. Crucially, this is not a homogeneous swarm of identical agents. It is a **heterogeneous network of experts**, where different agents are profiled and equipped to handle specific types of tasks.[3] This specialization is a key design principle, as a single, general-purpose agent trying to handle diverse tasks like planning, research, coding, and analysis would suffer from "skill dilution" and context-switching inefficiencies.[35]

Example Executor agent roles could include:

- **ResearchAgent:** Gathers information from the knowledge base or external sources (if permitted).[36]
- **CodingAgent:** Writes, tests, and debugs code in various languages.
- **DataAnalysisAgent:** Performs statistical analysis, generates visualizations, and interprets data.
- **ValidationAgent:** Checks the outputs of other agents for factual accuracy, logical consistency, or adherence to formatting rules.[37]
- **UserInteractionAgent:** Manages communication with a human user, asking for clarification or presenting final results.

## Core Agent Loop: Perceive, Plan, Act

Each Executor agent, regardless of its specialization, operates on a fundamental agentic loop.[38] It

**perceives** its assigned sub-task and the associated context provided by the Orchestrator. It then formulates a **micro-plan** for how to achieve this specific sub-goal. Finally, it **acts** on this plan, executing a sequence of operations until the sub-goal is achieved or it determines that it has failed.

## Intelligent Tool Selection and Execution

A core capability of every Executor agent is the ability to use tools. An agent's power comes from its ability to interact with systems beyond the LLM itself, such as databases, APIs, or a code interpreter.[27] The Executors use a

**ReAct (Reason+Act)** paradigm to guide their tool use.[27] For a given sub-task, the agent first

**reasons** about what it needs to do and which tool is best suited for the next step. It then generates the

appropriate **action** (e.g., an API call or a line of code). It executes this action and **observes** the result (e.g., the API response or a program error). This observation is then fed back into its context, and it begins the next cycle of reasoning about what to do next. This iterative loop allows the agent to perform complex, multi-step tasks that require interacting with the external world. The quality and clarity of the tool descriptions available to the agent are critical; a vague or misleading description can easily cause the agent to select the wrong tool, leading to failure.[28]

## The Communication Backbone: Asynchronous Messaging and Context Cohesion Protocols

For a large-scale, distributed system like the Infinitus Framework to function, the communication backbone that connects the Orchestrator and the Executors must be both highly efficient and robust. The challenges are twofold: managing the sheer volume of messages in a scalable way and ensuring that all agents maintain a coherent, shared understanding of the task context as it evolves.

### Asynchronous Communication

Given that agents operate at different speeds and on tasks of varying duration, synchronous communication (where one agent must wait for another to respond) is not viable as it would create massive bottlenecks. The framework therefore relies on an **asynchronous, event-driven architecture**.[29] Agents communicate by sending messages to queues or topics on a central message bus. This decouples the agents, allowing the Orchestrator to dispatch hundreds of tasks without waiting for each one to complete, and allowing Executors to work in parallel and report their results when ready. A

**blackboard pattern** is a particularly suitable model, where the Orchestrator posts task descriptions to a "tasks" board, and Executors pick them up. Once an Executor completes a task, it posts its results to a "results" board, which the Orchestrator monitors to update its overall plan state.[29] Using a durable, replayable log like Apache Kafka as the underlying technology for this blackboard can provide additional benefits of fault tolerance and state recovery.[30]

### Context Cohesion

One of the most difficult challenges in any multi-agent system is "context engineering"—effectively communicating the necessary context to each agent so it can perform its task correctly.[39] As a plan unfolds over hundreds of steps and involves dozens of agents, ensuring that context is passed accurately and

coherently is paramount. A single misinterpretation can cause the entire workflow to derail.

To address this, the Infinitus Framework mandates the implementation of a **Model Context Protocol (MCP)**.[6] MCP is a standardized specification for how context should be structured, packaged, and transmitted between agents. Rather than passing around unstructured blobs of text, MCP provides a formal schema for context payloads. A hypothetical MCP implementation would include [40]:

- **Semantic Tagging:** Metadata indicating the type of context (e.g., user_query, intermediate_result, error_log).
- **Versioning:** Timestamps or version numbers to handle state changes in long-running tasks and prevent agents from acting on stale information.
- **Source Attribution:** Clear identification of which agent generated the context.
- **Payload Optimization:** Mechanisms for compressing context to minimize network traffic.

By enforcing a standardized protocol like MCP, the framework ensures that context exchange between heterogeneous agents is reliable and unambiguous. This structured approach is essential for maintaining coherence in complex, multi-step, multi-agent operations and is a critical enabler of the system's overall robustness.[40]

## Core Capabilities for Infinite-Scale Operation

The architectural design of the Infinitus Framework is ambitious, envisioning a system that operates on a scale far beyond current capabilities. To make this vision a reality, the framework must be built upon three technological pillars that directly address the core limitations of today's AI: achieving a functionally infinite context window, enabling continuous and infinite activity through lifelong learning, and ensuring secure and reliable infinite delivery via offline deployment. The solutions to these challenges are not uniform across the system; they must be tailored to the specific needs of different agent types. The Orchestrator and the Executors, for instance, have fundamentally different memory requirements. The Orchestrator, managing a long, unfolding plan, needs a temporally coherent, sequential memory to track the plan's history. The Executors, performing narrow tasks, need a semantic, associative memory to recall specific facts and skills. A one-size-fits-all approach would be inefficient; therefore, a key design principle of the Infinitus Framework is the use of a heterogeneous memory architecture, matching the right technology to the right component.

## Achieving Infinite Context: Advanced Memory Architectures

The ability to process and retain information over arbitrarily long sequences is a prerequisite for the kind of complex, long-horizon planning the Orchestrator must perform. Standard Transformer models are

fundamentally constrained by their quadratic attention mechanism. The Infinitus Framework overcomes this by incorporating novel memory architectures that provide functionally infinite context with bounded computational resources.

## Compressive Memory: The Infini-Attention Mechanism

For the Orchestrator agent, whose primary need is to maintain a coherent history of a long and complex plan, the **Infini-attention** mechanism is an ideal solution.[7] Developed by Google researchers, Infini-attention augments the standard Transformer architecture with a

**compressive memory**. The mechanism works as a hybrid:

- For recent tokens within a local window, it uses the standard, highly effective dot-product attention mechanism.
- For tokens that fall outside this local window, instead of discarding their Key-Value (KV) states, it compresses them and stores them in a long-term, fixed-size memory store.[7]

During subsequent generation steps, the model's attention mechanism can query both the local context and this compressed long-term memory, allowing it to retrieve relevant information from the distant past. This effectively enables the model to handle infinitely long input sequences while keeping memory usage and computational costs bounded.[7] This architecture is perfectly suited for the Orchestrator, allowing it to maintain perfect recall of the entire problem-solving trajectory, from the initial user query to the results of every sub-task, without its context window ever overflowing.

## Human-Inspired Episodic Memory: The EM-LLM Approach

For the Executor agents, the memory requirement is different. They do not need to recall the entire plan history, but rather need to efficiently retrieve specific, semantically relevant pieces of knowledge to perform their specialized tasks. For this, the **EM-LLM** architecture, which is inspired by human episodic memory, is a more suitable choice.[8]

Instead of treating the context as a single, undifferentiated stream of tokens, EM-LLM first processes the information and segments it into discrete, coherent "events" or "episodes" in an online fashion.[8] This segmentation is driven by principles from cognitive science, using techniques like

**Bayesian surprise** to identify points where the information content changes significantly, thus marking a natural boundary between events.[8] When an Executor needs to access information, retrieval becomes a more efficient, two-stage process: first, it retrieves the most relevant

*episodes*, and then it performs fine-grained attention only within those selected episodes.[8] This dramatically reduces the search space and prevents attention from being diluted over thousands of irrelevant tokens. This approach is highly synergistic with the RAG pipeline, allowing an Executor to efficiently search through a vast local knowledge base by retrieving relevant "episodes" (e.g., specific sections of a technical manual or relevant past problem-solving cases) related to its current sub-task.

### Other Novel Frameworks

The field of long-context memory is evolving rapidly, and other promising techniques could also be integrated. **InfiniPot** is a training-free framework that uses a process called Continual Context Distillation (CCD) to intelligently compress the KV-cache when it approaches its limit, retaining the most essential information.[43]

**EdgeInfinite** proposes a trainable memory-gating module designed specifically to enable long-context processing on resource-constrained edge devices.[44] A mature Infinitus system could dynamically select the most appropriate memory management technique based on the task and the available hardware.

## Enabling Infinite Activity: Continual Learning and Adaptation

For the Infinitus Framework to operate 24/7 and truly improve over time, it must be capable of **continual learning (CL)**, also known as lifelong learning. This is the ability to acquire new knowledge and skills sequentially from a continuous stream of data without needing to be taken offline for retraining and, crucially, without catastrophically forgetting what it has already learned.[2]

### A Taxonomy of Continual Learning (CL) Methods

Research in continual learning has produced three main paradigms for mitigating catastrophic forgetting, each inspired by mechanisms in the human brain.[2] The Infinitus Framework would employ a hybrid of these methods, tailored to the different learning needs of the Orchestrator and the Executors.

1. **Replay-based Methods:** Inspired by hippocampal memory replay, these methods store a small subset of data from past tasks in a "memory buffer".[2] When a new task is learned, the model is trained on a mix of the new data and the replayed data from the buffer. This directly reminds the model of old tasks, preventing it from forgetting them. This approach is well-suited for the Orchestrator, which could replay the execution traces of past successful problem decompositions to

reinforce and refine its high-level planning strategies.

2. **Regularization-based Methods:** These methods add a regularization term to the model's loss function during training. This term penalizes large changes to the model parameters that were identified as being critical for performance on previous tasks.[2] This is analogous to synaptic consolidation in the brain, protecting important learned connections. This could be used by both Orchestrator and Executor agents to ensure that learning a new type of task does not degrade performance on existing ones.
3. **Architecture-based Methods:** These methods prevent interference by isolating task-specific knowledge within the model's architecture. This can be done by dynamically expanding the network to add new modules for new tasks, or more efficiently, by using Parameter-Efficient Fine-Tuning (PEFT) techniques like Low-Rank Adaptation (LoRA). With LoRA, the main weights of the pre-trained model are kept frozen, and a small number of new, trainable "adapter" weights are added for each new task or skill.[2] This is a perfect fit for the Executor agents. For example, a CodingAgent could learn to work with a new programming language by adding a new LoRA adapter, without any risk of forgetting the languages it already knows.

## Mitigating Catastrophic Forgetting in a Multi-Agent Context

By combining these approaches, the Infinitus Framework can achieve robust, system-wide continual learning. When the system learns a new skill, it would not involve retraining the entire system. Instead, a new specialized Executor agent might be instantiated, or an existing Executor might be updated with a new architecture-based module (like a LoRA adapter). The Orchestrator would then update its agent registry to be aware of this new capability. It would learn how to incorporate this new skill into its plans through its own replay-based learning, studying new successful execution traces that utilize the new capability. This multi-faceted approach provides a scalable and robust solution to the challenge of lifelong learning in a complex, distributed system.[9]

## Ensuring Infinite Delivery: Offline Deployment and Knowledge Grounding

The final pillar of infinite-scale operation is the ability to deliver results reliably and securely, independent of external networks. The Infinitus Framework is designed for **complete offline deployment**, ensuring data privacy, security, and operational resilience.[10] This is achieved through a combination of on-device model execution and a powerful mechanism for grounding the system in a user's private data.

## On-Device Model Execution and Local API Servers

The entire Infinitus system—the Orchestrator, the Executor agents, and the underlying LLMs—is intended to run on local, user-controlled hardware. This requires a reasonably powerful machine (e.g., a modern multi-core CPU, sufficient RAM, and one or more GPUs), but it completely eliminates the privacy risks associated with sending data to third-party cloud services.[10]

To facilitate communication between the agents and the models, the framework utilizes **local inference servers**. Software like Ollama or Jan.ai allows users to download and run open-source LLMs (e.g., Llama 3, Mistral) on their own machines.[10] These tools expose an OpenAI-compatible API endpoint. This is a crucial feature, as it allows the agent framework (e.g., LangChain, Griptape) to interact with the local models using the same code and protocols as it would for a cloud-based model like GPT-4.[46] This provides the benefits of a familiar development paradigm while ensuring all processing and data remain within the user's secure, air-gapped environment.

## Retrieval-Augmented Generation (RAG) for Custom Knowledge Integration

Offline operation means the system cannot access the live web for information. Its knowledge is therefore bounded by what is provided to it. **Retrieval-Augmented Generation (RAG)** is the key technology that allows the system to be grounded in a vast, private, and user-curated knowledge base.[10] The RAG pipeline consists of two main stages [48]:

1. **Ingestion and Indexing:** This is a one-time, offline preprocessing step. The user provides a collection of documents (e.g., technical manuals, research papers, internal wikis, project histories). The RAG system processes these documents by:
    - **Splitting:** The documents are divided into smaller, manageable chunks of text.
    - **Embedding:** Each chunk is passed through an embedding model (which also runs locally) to convert it into a numerical vector that captures its semantic meaning.
    - **Storing:** These vectors and their corresponding text chunks are stored in a local **vector database** (e.g., ChromaDB, FAISS).[10] This creates a searchable index of the entire knowledge base.
2. **Retrieval and Generation:** This happens in real-time during task execution. When an Executor agent needs information to complete its sub-task, it:
    - **Retrieves:** It takes its current query (e.g., "What is the standard procedure for deploying the X-7 module?") and converts it into a vector. It then performs a similarity search against the vector database to find the most relevant text chunks from the original documents.
    - **Augments:** It takes these retrieved chunks of text and injects them into its prompt to the LLM, along with the original query.
    - **Generates:** The LLM then generates its response based not only on its pre-trained knowledge

but also on the specific, highly relevant, and factual context provided from the user's private documents.

This RAG pipeline is fundamental to the framework's utility. It ensures that the system's responses are accurate, grounded in the user's specific domain, up-to-date with the provided knowledge, and far less prone to the kind of "hallucinations" that plague un-grounded models.[47] It allows the Infinitus system to become a true expert in any domain for which it is given a knowledge base.

## Systemic Challenges and Mitigation Strategies

The creation of a system as complex and powerful as the Infinitus Framework introduces a new class of systemic challenges that go beyond the performance of any single component. The interactions between dozens, hundreds, or even thousands of autonomous, continually learning agents create the potential for emergent behaviors that are difficult to predict and control. These challenges related to error propagation, security, scalability, and verifiability are not independent problems but are deeply intertwined facets of a single, overarching meta-problem: **managing systemic complexity**. An intervention designed to solve one issue can often have unintended, negative consequences in another domain. For example, adding more ValidationAgents to mitigate error propagation inevitably increases the communication load and computational overhead, exacerbating the scalability challenge. Similarly, adopting a sparser communication topology to improve scalability might slow down the detection of critical errors. Therefore, designing a robust Infinitus system is not about finding a perfect, "silver bullet" solution for each challenge, but about understanding these interdependencies and finding an optimal **trade-off point** within the multi-dimensional space of safety, performance, and cost.

## Error Propagation and Correction in a Distributed Network

In any multi-agent system where the output of one agent serves as the input for another, the risk of **cascading error propagation** is a primary concern.[37] LLMs are inherently non-deterministic and are known to confabulate or "hallucinate" plausible but incorrect information.[50] In the Infinitus Framework, a minor error generated by a single Executor agent—a slightly wrong coordinate, a flawed piece of code, a misremembered fact—can be passed to the next agent in the chain. Because agents, by default, tend to trust the messages they receive from their peers, this error is not only propagated but can be amplified as it moves through the network, potentially leading to a complete derailment of the overall task.[37] This danger is magnified in a continually learning system, where a flawed output that is part of a successful task completion might be incorporated into a model's weights as a positive example, permanently embedding the error into the system's knowledge.

Research into communication topologies in MAS suggests that moderately sparse graphs can achieve an optimal balance, effectively suppressing the propagation of errors while still preserving the diffusion of beneficial information.[51] However, topology alone is insufficient. A multi-pronged mitigation strategy is required:

- **Evaluator Agents:** The most direct solution is to implement dedicated ValidationAgents or EvaluatorAgents within the network.[37] The sole function of these agents is to act as quality control gates. They intercept the outputs of other agents and validate them against a set of rules, constraints, or factual knowledge before allowing them to be passed on. This could involve checking for proper data formatting, validating claims against the RAG knowledge base, or even running unit tests on generated code. This approach institutionalizes a "zero-trust" policy between agents and contains errors at their source. This is a form of collaborative, multi-agent error detection, where agents check the work of other agents.[52]
- **Self-Correction and Reflection:** Before finalizing its output, an agent can employ a "reflection" step, a mechanism where it prompts itself to critique its own work, check its reasoning, and identify potential flaws.[1] This internal review process can catch many errors before they ever leave the agent.
- **Stateful Execution and Checkpoints:** Given that agents can run for long periods and errors can occur due to transient system failures, the framework must be designed for resilient execution. This means durably saving the state of the system at regular **checkpoints**.[28] If a catastrophic error occurs, the system does not need to restart from the beginning; it can be rolled back to the last known good state and resume execution, saving significant time and computational resources. The system can even inform the agent that a tool has failed, allowing the agent to adapt its plan accordingly.[28]
- **Human-in-the-Loop (HITL) Validation:** For particularly high-stakes or ambiguous tasks, the system must have the ability to pause and request confirmation or clarification from a human user. This ensures that a human expert remains in ultimate control of critical decision points.[27]

To provide a more structured understanding of this challenge, the following table outlines a taxonomy of common error types in a MAS like the Infinitus Framework and maps them to their most effective mitigation strategies.

| Error Type | Description | Point of Origin | Primary Mitigation Strategy | |
|---|---|---|---|---|
| **Factual Hallucination** | Agent generates plausible but factually incorrect information that is not grounded in its knowledge base or observations. | Executor Agent | Retrieval-Augmented Generation (RAG) to ground all outputs in a trusted knowledge base.[47] | ValidationAgent to fact-check claims against the knowledge base.[37] |

| | | | |
|---|---|---|---|
| **Tool Misuse** | Agent selects an incorrect tool for the sub-task or uses the correct tool with improper parameters or syntax. | Executor Agent | Clear and unambiguous tool descriptions.[28] Self-correction loop where the agent analyzes tool errors and retries.[1] |
| **Planning Fallacy** | Orchestrator creates an illogical, inefficient, or incomplete decomposition tree for a given task. | Orchestrator Agent | Continual learning with reinforcement on successful vs. unsuccessful execution traces. Formal verification of the underlying HTN planner structure.[54] |
| **Coordination Deadlock** | Two or more agents enter a state where they are waiting for each other to act, causing the system to hang. | Communication Backbone | Asynchronous, event-driven communication protocols to decouple agents.[29] Timeouts and explicit error handling in the communication layer. |
| **State Inconsistency** | An agent acts on outdated information because it did not receive a timely update from another agent. | Communication Backbone | Implementation of a robust Context Cohesion Protocol (e.g., MCP) with versioning and timestamps.[40] Use of a single source-of-truth event log.[30] |
| **Cascading** | A minor error | System-wide | A combination of |

| | | | |
|---|---|---|---|
| **Error** | from one agent is passed through the system and amplified, leading to a major downstream failure. | | all the above: ValidationAgents for containment, self-correction for prevention, and checkpoints for recovery.[28] |

## Security and Trust in Offline, User-Grounded Systems

While designing the Infinitus Framework for offline operation inherently mitigates a large class of security threats associated with cloud computing, such as data breaches in transit or on third-party servers [10], it introduces a new and critical attack surface: the

**user-provided knowledge base**.

The primary threat in this air-gapped environment becomes **data poisoning**.[55] An adversary could intentionally introduce malicious, biased, or subtly incorrect information into the documents that are fed into the RAG ingestion pipeline. Because the system is designed to trust this knowledge base as its source of truth, a poisoned knowledge base could be used to manipulate the system's behavior from the inside out. For example, an attacker could insert a document into a corporate knowledge base that describes a "new standard procedure" which, when executed by a

CodingAgent, introduces a security vulnerability into the company's software. Detecting such subtle manipulations is a significant challenge.

Other threats, such as model inversion or membership inference attacks, where an attacker tries to recover sensitive training data by carefully querying a model, are still relevant even for local models.[55] An attacker with access to the system could potentially reconstruct parts of the proprietary knowledge base.

Mitigating these threats requires a security-in-depth approach:

- **Secure Ingestion Pipeline:** The RAG ingestion process cannot blindly trust all input documents. It must include a **sanitization and validation** layer that scans documents for known malware, flags suspicious content, and perhaps even uses another AI agent to cross-reference claims against a known-good subset of the data.[55]
- **Principle of Least Privilege:** Even within the offline system, access should be controlled. Agents should only be granted access to the tools and data segments that are strictly necessary for their defined role. For example, a UserInteractionAgent should not have access to the code execution tool.[56]

- **Threat Modeling:** Organizations deploying an Infinitus system must conduct rigorous threat modeling that covers the entire AI/ML supply chain. This includes not just the software components but, most importantly, the provenance and trustworthiness of the data sources used to build the knowledge base.[55]

### Scalability, Latency, and Computational Overhead

A defining characteristic of multi-agent systems is their potential for high computational and communication costs. Empirical analysis has shown that multi-agent systems can use up to 15 times more tokens (and thus, more computational resources) than a simple chat interaction to solve the same problem.[28] This overhead stems from the communication between agents, the reasoning steps performed by each agent, and the potential for redundant work.

The primary scalability challenges are:

- **Communication Overhead:** In a naive, fully connected network, the number of potential communication channels grows exponentially with the number of agents. This can quickly overwhelm network resources and create severe performance bottlenecks, leading to sluggish response times or even system-wide deadlocks.[58]
- **Resource Contention:** In a large-scale deployment, multiple agents running concurrently on the same hardware will compete for shared resources like CPU, GPU, and memory. This can lead to emergent performance degradation, where agents unknowingly "starve" each other of resources, creating bottlenecks that are difficult to diagnose by monitoring any single agent in isolation.[59]

Mitigation strategies must focus on architectural efficiency:

- **Efficient Communication Topologies:** Instead of a fully connected "all-to-all" graph, the framework should employ more efficient communication topologies. Architectures like a **star** (all Executors communicate only with the central Orchestrator), a **ring**, or a **moderately sparse graph** can dramatically reduce the communication load while maintaining effective coordination.[51]
- **Asynchronous, Event-Driven Architecture:** As discussed previously, decoupling agents via message queues and an event-driven design is critical for scalability. It allows for high levels of parallelism and prevents the entire system from being blocked by a single slow agent.[29]
- **Task-Value Justification:** Given the inherent costs, it is crucial to recognize that multi-agent systems are not the right solution for every problem. They are best suited for high-value, complex tasks where the significant performance gains from parallelization and specialization justify the increased computational expense.[28] The system should be applied to problems that are either intractable for a single agent or where the speed-up provides a clear economic or strategic advantage.

## Formal Verification and Explainable AI (XAI) for Systemic Transparency

A system that is autonomous, distributed, and continually learning presents a profound challenge for trust and accountability. How can we be confident that such a system will behave as intended, especially in safety-critical applications? The answer lies in a dual approach: using **formal verification** to provide mathematical guarantees about core components and employing **Explainable AI (XAI)** to make the system's dynamic behavior transparent to human operators.

- **Formal Verification:** This is a set of techniques from theoretical computer science that use mathematical proof to verify that a system model satisfies a given specification (e.g., proving that a system can never enter a deadlock state).[54] Verifying an entire open, adaptive MAS is generally considered undecidable, meaning it is impossible to create an algorithm that can do it for all cases.[17] However, formal verification can and should be applied to the most critical, static components of the Infinitus Framework. For example, one could formally verify the communication protocol to prove it is free of race conditions, or verify the underlying logic of the HTN planner to ensure its decompositions are always sound.[54] This provides a bedrock of reliability upon which the more dynamic, learning-based components can be built.
- **Explainable AI (XAI):** Since we cannot formally verify the entire system's behavior, we must make its reasoning process transparent and understandable to its human users.[64] XAI aims to open up the "black box" of AI decision-making. In the Infinitus Framework, XAI is crucial for building trust and enabling effective debugging. A key technique that can be implemented is
  **Layered Prompting**, where the system is designed to break down its reasoning into a hierarchical series of justifiable steps.[53] The Orchestrator, for instance, would not just produce a final answer. It would be able to output its entire decomposition tree, explaining
  *why* it chose to break the problem down in a particular way, which agents it assigned to each sub-task, and how it synthesized their results. This structured, step-by-step explanation allows a human to trace the system's logic from the high-level goal to the final output, fostering trust and providing an invaluable tool for diagnosing failures.[53]

## Applications and Broader Implications

The Infinitus Framework, by integrating perpetual activity, unbounded context, and secure autonomous operation, represents a significant leap in AI capability. Its potential applications are transformative, promising to reshape entire industries and aspects of daily life. However, the very characteristics that make this framework so powerful—its autonomy, adaptability, and capacity for complex planning—also introduce profound ethical challenges that demand careful consideration and proactive governance. The core capability of recursive goal decomposition is fundamentally value-neutral; the same mechanism that could be tasked with "discovering a new material for carbon capture" could also be tasked by a malicious actor with "designing a social engineering campaign to destabilize a market." This duality means that the

most critical focus for responsible development is not just on aligning the model to be "good," but on building robust, unbreakable constraints around the goal-setting process itself, ensuring that the system's immense power can only be directed towards legitimate and beneficial ends.

## Potential Use Cases

The ability to automate complex, long-horizon, knowledge-intensive tasks opens up a vast design space for new applications.

### Automating Scientific Discovery

The framework is ideally suited to function as an "AI Scientist," a system capable of automating the entire scientific research lifecycle.[66] Grounded with a knowledge base containing the entirety of a scientific domain's literature, an Infinitus system could:

- **Automate Systematic Literature Reviews (SLRs):** It could autonomously perform the time-consuming process of identifying, filtering, and synthesizing thousands of research papers to establish the current state of the art.[68]
- **Generate and Test Hypotheses:** A ResearchAgent could analyze the literature and experimental data to formulate novel, plausible hypotheses. The Orchestrator could then decompose the goal of "test this hypothesis" into a series of steps for other agents to design experiments, write code to run simulations on high-performance computing clusters, and have a DataAnalysisAgent interpret the results.[66]
- **Accelerate Breakthroughs:** This closed-loop automation of hypothesis, experimentation, and interpretation could dramatically accelerate the pace of discovery in fields like medicine, materials science, and astrobiology, where AI systems are already being used to discover novel drug combinations and materials in a fraction of the time it would take humans.[66]

### Autonomous Business Operations

The Infinitus Framework could serve as the operating system for a fully autonomous business, where strategic goals are translated into coordinated actions by a network of AI agents.[70] This moves beyond simple task automation to autonomous process management.[38]

- **Strategic Decision Support:** An agent grounded in real-time market data, financial reports, and

internal operational metrics could act as an executive advisor, providing complex risk assessments and strategic forecasts.[70]

- **Automated Supply Chain Management:** The system could autonomously manage an entire supply chain, from forecasting demand and optimizing inventory to negotiating with supplier agents and rerouting logistics in response to real-time disruptions.[58]
- **Continuous Compliance and Auditing:** A team of agents could monitor all business transactions in real-time, flagging anomalies, ensuring adherence to complex regulatory requirements, and generating audit reports automatically.[70] This would transform compliance from a periodic, manual process into a continuous, automated function.

### Proactive Personal Assistants

The framework could power the next generation of truly intelligent personal assistants, moving beyond simple command-and-response to become proactive, context-aware companions.[71] With its infinite context (long-term memory) and continual learning capabilities, such an assistant could:

- **Maintain Lifelong Context:** It would remember every past interaction, preference, and important detail, allowing it to provide deeply personalized and coherent support without needing to be reminded of past conversations.[26]
- **Manage Complex Personal Logistics:** It could handle layered, multi-step requests like, "Find a highly-rated Italian restaurant near my 7 PM meeting that can accommodate a vegetarian, book a table for three, and add it to my calendar," by decomposing the task and executing each step.[71]
- **Act Proactively:** By learning a user's routines and goals, the assistant could anticipate needs, sending helpful reminders ("Your flight is in 3 hours and there is heavy traffic on the way to the airport, you should leave now"), managing conflicting appointments, and proactively organizing information relevant to the user's upcoming activities.[26]

### The Ethical Landscape: Accountability, Bias, Deception, and Job Displacement

The immense potential of the Infinitus Framework is matched by the gravity of its ethical implications. The deployment of powerful, autonomous systems necessitates a rigorous examination of these challenges.

- **Accountability and Responsibility:** When a complex, distributed system of autonomous agents causes harm, establishing accountability becomes incredibly difficult. Is the fault with the developer who wrote the code, the organization that deployed the system, the user who gave it a vague goal, or the agent that made a specific decision? This ambiguity in the chain of responsibility is a critical, unresolved legal and ethical issue.[73] Without transparency into the system's decision-making process,

assigning blame is nearly impossible, highlighting the need for the XAI mechanisms discussed earlier.[74]

- **Bias and Manipulation:** AI systems learn from data, and if that data reflects existing societal biases, the agents will learn, perpetuate, and even amplify those biases.[73] A hiring agent trained on historical data might learn to discriminate against certain demographic groups. Furthermore, there is a significant risk of agents learning to be
  **deceptive or manipulative** to achieve their goals. Research has shown that advanced models are capable of "scheming" to accomplish objectives, and an agent might learn that misleading a user is the most efficient path to fulfilling its assigned task.[74]
- **Security and Misuse:** Powerful autonomous agents are a quintessential dual-use technology. The same capabilities that allow the system to automate scientific research could be hijacked by malicious actors to automate cyberattacks, create hyper-realistic deepfakes for disinformation campaigns at an unprecedented scale, or design sophisticated social engineering schemes.[73] The prospect of a fully autonomous system capable of writing and executing its own code without constraints is particularly concerning, as it could potentially override human control and pursue harmful goals.[77]
- **Job Displacement:** A system designed to automate complex cognitive tasks will inevitably have a profound impact on the labor market. Roles that are heavily reliant on data entry, administration, analysis, and even coding are highly susceptible to automation by such a system.[78] While new jobs related to managing and overseeing these AI systems will be created, the scale of potential displacement could lead to significant societal disruption, requiring massive, coordinated efforts in workforce retraining and social safety net reform.[78]

## A Framework for Responsible Development and Deployment

Navigating this complex ethical landscape requires embedding principles of responsibility directly into the design and deployment lifecycle of these systems. The following principles should be considered mandatory for any organization developing technology based on the Infinitus Framework:

- **Human-in-the-Loop by Design:** For any decision that is high-stakes, irreversible, or ethically ambiguous, the system must be designed to pause and require explicit approval from a human operator. Full autonomy should be reserved for tasks where the cost of an error is low.[73]
- **Graduated Autonomy:** Systems should not be deployed with full autonomy from the outset. Instead, autonomy should be **graduated**, beginning in a highly constrained mode with significant human oversight. As the system demonstrates reliability and builds trust over time, its level of autonomy can be incrementally increased.[77]
- **Ethical Guardrails and Alignment:** The system's goals must be aligned with human values. This can be achieved through technical methods like Reinforcement Learning from Human Feedback (RLHF), or by designing the system to adhere to explicit, human-written policy documents, such as corporate ethics guidelines or government regulations.[75]

- **AI Security Compliance Programs:** The risks are too great to rely on self-regulation alone. This paper advocates for the creation of formal **AI Security Compliance programs**, analogous to existing compliance frameworks in critical industries like finance (PCI compliance) or healthcare (HIPAA). Such programs would mandate a set of best practices for securing AI systems, including rigorous threat modeling, data provenance tracking, and the implementation of robust attack response plans, enforced by appropriate regulatory bodies.[57]

## Conclusion and Future Work

### Summary of the Infinitus Framework

This paper has introduced the Infinitus Framework, a conceptual architecture for a new generation of autonomous AI systems designed for perpetual operation with functionally infinite context and secure, offline delivery. By synthesizing research from multi-agent systems, recursive task decomposition, advanced memory architectures, continual learning, and Retrieval-Augmented Generation, the framework provides a blueprint for a system that can tackle complex, long-horizon problems with unprecedented autonomy and scale.

The core of the framework is the **Orchestrator-Executor model**, where a central Orchestrator agent performs self-improving, recursive decomposition of complex goals, distributing the resulting sub-tasks to a heterogeneous network of specialized Executor agents. This process is enabled by key technological pillars: a **heterogeneous memory architecture** (combining compressive and episodic memory) to overcome context limitations; a **hybrid continual learning strategy** (combining replay, regularization, and architecture-based methods) to enable infinite activity without catastrophic forgetting; and a design for **offline deployment** grounded in a private user knowledge base via RAG to ensure security and reliability. The framework is not merely a collection of components, but a cohesive system where these technologies work synergistically, creating emergent properties like a system-wide self-improvement loop fueled by the very act of problem-solving.

### Key Hurdles to Realization and Directions for Future Research

While the Infinitus Framework is conceptually sound and grounded in existing research, its practical realization faces significant hurdles that represent key directions for future work.

- **Theoretical Hurdles:** The emergent behavior of a large-scale, adaptive MAS is not yet well

understood. A more **formal theory of emergent computation** in these systems is needed to predict and control their collective behavior. Furthermore, developing robust and scalable methods for the **formal verification of continually learning systems** remains a major open problem in computer science. Proving properties about a system whose logic and parameters are constantly changing is a profound theoretical challenge.

- **Engineering Hurdles:** Several significant engineering challenges must be overcome. Building a truly efficient, low-latency, and scalable **asynchronous communication backbone** capable of handling millions of messages per second is a non-trivial software engineering task. The development and standardization of a universal **Model Context Protocol (MCP)** and associated tool-use protocols are essential for enabling seamless interoperability between agents from different developers. Finally, the sheer **hardware requirements** for running the entire system—including multiple powerful LLMs and a massive vector database—locally with acceptable latency remain a significant barrier to widespread adoption.
- **Ethical Hurdles:** The most pressing challenges are ethical. Establishing clear international norms and **legal frameworks for accountability** in the event of harm caused by autonomous AI is a task that will require collaboration between technologists, ethicists, policymakers, and the legal community. Perhaps the most critical area for future research is in **"goal security"**—developing provably robust mechanisms to prevent a powerful autonomous system from being tasked with malicious or harmful goals by unauthorized users.
- **Future Work:** This paper calls for focused research in several key areas to advance the concepts presented. A primary focus should be the development and empirical evaluation of the proposed **hybrid HTN/HRL planner** for the Orchestrator, combining the structure of classical planning with the adaptability of reinforcement learning. Another critical area is conducting large-scale empirical studies on the complex trade-offs between **communication topology, error propagation rates, and overall system performance**. Finally, further research into lightweight, efficient, and training-free methods for **infinite context management** and **continual learning** will be essential to making systems like the Infinitus Framework practical and accessible.

## Works cited


1. A Survey on Large Language Model based Autonomous ... - arXiv, accessed on July 14, 2025, http://arxiv.org/pdf/2308.11432
2. A Comprehensive Survey on Continual Learning in ... - arXiv, accessed on July 14, 2025, https://www.arxiv.org/pdf/2506.13045v1
3. What are AI agents? Definition, examples, and types | Google Cloud, accessed on July 14, 2025, https://cloud.google.com/discover/what-are-ai-agents
4. (PDF) Multi-Agent Systems: A Survey About Its Components ..., accessed on July 14, 2025, https://www.researchgate.net/publication/381151924_Multi-agent_Systems_A_survey_about_its_components_framework_and_workflow
5. Multi-agent system - Wikipedia, accessed on July 14, 2025, https://en.wikipedia.org/wiki/Multi-agent_system
6. Multi-Agent AI Systems (MAS) And The Path To Organizational Success - Forbes, accessed on July 14, 2025, https://www.forbes.com/councils/forbesbusinesscouncil/2025/06/16/multi-agent-ai-systems-mas-and-the-path-to-organizational-success/

7. Infinite Context Length in LLMs — The Next Big Advantage in AI | by Aloy Banerjee | Medium, accessed on July 14, 2025, https://medium.com/@aloy.banerjee30/infinite-context-length-in-llms-the-next-big-advantage-in-ai-2550e9e6ce9b
8. Human-inspired Episodic Memory for Infinite Context LLMs ..., accessed on July 14, 2025, https://openreview.net/forum?id=BI2int5SAC
9. Advancing autonomy through lifelong learning: a survey of autonomous intelligent systems, accessed on July 14, 2025, https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2024.1385778/full
10. Offline AI Made Easy: How to Run Large Language Models Locally, accessed on July 14, 2025, https://blog.marketingdatascience.ai/offline-ai-made-easy-how-to-run-large-language-models-locally-1dd3bbbf214e
11. How developers are using offline AI tools for air-gapped security | by Pieces - Medium, accessed on July 14, 2025, https://pieces.medium.com/how-developers-are-using-offline-ai-tools-for-air-gapped-security-f74a4ad4c7e2
12. You can do what with Pieces AI when offline?, accessed on July 14, 2025, https://pieces.app/blog/offline-ai
13. Multi-Agent Systems and Coordination | by Volodymyr Pavlyshyn | Jul, 2025 | Medium, accessed on July 14, 2025, https://medium.com/@volodymyrpavlyshyn/multi-agent-systems-and-coordination-645a317b4c61
14. Generating a Low-code Complete Workflow via Task Decomposition and RAG - arXiv, accessed on July 14, 2025, https://arxiv.org/html/2412.00239v1
15. How task decomposition and smaller LLMs can make AI more ..., accessed on July 14, 2025, https://www.amazon.science/blog/how-task-decomposition-and-smaller-llms-can-make-ai-more-affordable
16. Prompting LLMs to Solve Complex Tasks: A Review - International Journal of Information Technology, accessed on July 14, 2025, https://intjit.org/journal/download/down.php?file=/29/1/291_3.pdf
17. Hierarchical task network - Wikipedia, accessed on July 14, 2025, https://en.wikipedia.org/wiki/Hierarchical_task_network
18. Hierarchical Task Network (HTN) Planning in AI - GeeksforGeeks, accessed on July 14, 2025, https://www.geeksforgeeks.org/artificial-intelligence/hierarchical-task-network-htn-planning-in-ai/
19. Hierarchical Task Network (HTN) Planning, accessed on July 14, 2025, https://pages.mtu.edu/~nilufer/classes/cs5811/2012-fall/lecture-slides/cs5811-ch11b-htn.pdf
20. Building Exospecies AI: Hierarchical Task Networks Overview - Eric Zinda Blog, accessed on July 14, 2025, https://blog.inductorsoftware.com/blog/htnoverview
21. Decomposing user-defined tasks in a reinforcement learning setup using TextWorld, accessed on July 14, 2025, https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2023.1280578/full
22. Hierarchical Reinforcement Learning, Sequential Behavior, and the Dorsal Frontostriatal System - PMC, accessed on July 14, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC9274316/
23. Hierarchical Reinforcement Learning - Towards Data Science, accessed on July 14, 2025, https://towardsdatascience.com/hierarchical-reinforcement-learning-56add31a21ab/
24. Hierarchical Reinforcement Learning (HRL): Breaking Down Complex Tasks | by Hey Amit, accessed on July 14, 2025, https://medium.com/@heyamit10/hierarchical-reinforcement-learning-hrl-breaking-down-complex-tasks-d9798e49c782

25. Professional Agents - Evolving Large Language Models into Autonomous Experts with Human-Level Competencies - arXiv, accessed on July 14, 2025, https://arxiv.org/html/2402.03628v1
26. AI Memory: How Smart Assistants Learn and Retain Data - Tanka, accessed on July 14, 2025, https://www.tanka.ai/blog/posts/ai-memory
27. What Are AI Agents? | IBM, accessed on July 14, 2025, https://www.ibm.com/think/topics/ai-agents
28. How we built our multi-agent research system - Anthropic, accessed on July 14, 2025, https://www.anthropic.com/engineering/built-multi-agent-research-system
29. How do multi-agent systems handle asynchronous communication? - Milvus, accessed on July 14, 2025, https://milvus.io/ai-quick-reference/how-do-multiagent-systems-handle-asynchronous-communication
30. A Distributed State of Mind: Event-Driven Multi-Agent Systems | by Sean Falconer | Medium, accessed on July 14, 2025, https://seanfalconer.medium.com/a-distributed-state-of-mind-event-driven-multi-agent-systems-226785b479e6
31. Post on Self Improving LLMs Through Recursive Problem Decomposition (LADDER) | by Dr. Vijay Srinivas Agneeswaran | Jul, 2025, accessed on July 14, 2025, https://a-vijaysrinivas.medium.com/post-on-self-improving-llms-through-recursive-problem-decomposition-ladder-eed13f899a18
32. ADaPT: As-Needed Decomposition and Planning ... - GitHub Pages, accessed on July 14, 2025, https://allenai.github.io/adaptllm/
33. LADDER: Self-Improving LLMs Through Recursive Problem Decomposition - arXiv, accessed on July 14, 2025, https://arxiv.org/html/2503.00735v1
34. LADDER: Self-Improving LLMs Through Recursive Problem Decomposition - arXiv, accessed on July 14, 2025, https://arxiv.org/html/2503.00735v3
35. Scaling Intelligence: Multi-Agent Design Patterns for Efficient and Specialized AI Systems | by Himank Jain | Medium, accessed on July 14, 2025, https://medium.com/@himankvjain/scaling-intelligence-multi-agent-design-patterns-for-efficient-and-specialized-ai-systems-fb6503b71726
36. Build AI Agents for Research - Relevance AI, accessed on July 14, 2025, https://relevanceai.com/function/research
37. 7 Multi-Agent Debugging Challenges Every AI Team Faces | Galileo, accessed on July 14, 2025, https://galileo.ai/blog/debug-multi-agent-ai-systems
38. With autonomous problem-solving, agentic AI will upend what you consider work, accessed on July 14, 2025, https://blogs.lse.ac.uk/businessreview/2025/02/11/with-autonomous-problem-solving-agentic-ai-will-upend-what-you-consider-work/
39. How and when to build multi-agent systems - LangChain Blog, accessed on July 14, 2025, https://blog.langchain.com/how-and-when-to-build-multi-agent-systems/
40. Implementing Model Context Protocol in Autonomous Multi-Agent Systems - Technical Architecture and Performance Optimization | Subhadip Mitra, accessed on July 14, 2025, https://subhadipmitra.com/blog/2025/implementing-model-context-protocol/
41. Scaling Large Language Models (LLMs) for Infinite Context: A ..., accessed on July 14, 2025, https://www.ampcome.com/articles/how-to-scale-large-language-models-llms-to-infinite-context
42. EM-LLM: Human-Inspired Episodic Memory for Infinite Context LLMs | Hacker News, accessed on July 14, 2025, https://news.ycombinator.com/item?id=43943928

43. InfiniPot: Infinite Context Processing on Memory-Constrained LLMs - arXiv, accessed on July 14, 2025, https://arxiv.org/html/2410.01518v1
44. EdgeInfinite: A Memory-Efficient Infinite-Context Transformer for Edge Devices - arXiv, accessed on July 14, 2025, https://arxiv.org/abs/2503.22196
45. Continual Learning: Methods and Application - neptune.ai, accessed on July 14, 2025, https://neptune.ai/blog/continual-learning-methods-and-application
46. Jan: Open source ChatGPT-alternative that runs 100% offline - Jan, accessed on July 14, 2025, https://jan.ai/
47. Implementing Retrieval-Augmented Generation (RAG) with Amazon Bedrock Knowledge Bases | by Rick Hightower | Medium, accessed on July 14, 2025, https://medium.com/@richardhightower/implementing-retrieval-augmented-generation-rag-with-amazon-bedrock-knowledge-bases-362cb54c522d
48. Introducing React Native RAG: Local & Offline Retrieval-Augmented ..., accessed on July 14, 2025, https://blog.swmansion.com/introducing-react-native-rag-fbb62efa4991
49. Retrieval-Augmented Generation (RAG) from basics to advanced | by Tejpal Kumawat, accessed on July 14, 2025, https://medium.com/@tejpal.abhyuday/retrieval-augmented-generation-rag-from-basics-to-advanced-a2b068fd576c
50. Code versus Model in Multi-Agentic Systems | by Decision AI | May, 2025 - Medium, accessed on July 14, 2025, https://medium.com/@evolutionmlmail/code-versus-model-in-multi-agentic-systems-e33cf581e32b
51. [2505.23352] Understanding the Information Propagation Effects of Communication Topologies in LLM-based Multi-Agent Systems - arXiv, accessed on July 14, 2025, https://arxiv.org/abs/2505.23352
52. Harnessing Diverse Perspectives: A Multi-Agent Framework for Enhanced Error Detection in Knowledge Graphs - arXiv, accessed on July 14, 2025, https://arxiv.org/html/2501.15791v1
53. (PDF) Explainable AI in Multi-Agent Systems: Advancing ..., accessed on July 14, 2025, https://www.researchgate.net/publication/388835453_Explainable_AI_in_Multi-Agent_Systems_Advancing_Transparency_with_Layered_Prompting
54. Formal Modeling, Specification, and Verification of Multi-Agent Systems - Inria, accessed on July 14, 2025, https://smimram.gitlabpages.inria.fr/lhc/slides/2025/malvone.pdf
55. Threat Modeling AI/ML Systems and Dependencies | Microsoft Learn, accessed on July 14, 2025, https://learn.microsoft.com/en-us/security/engineering/threat-modeling-aiml
56. Building a Multi-Agent System for Automatic Code Error Detection from Screenshots, accessed on July 14, 2025, https://www.analyticsvidhya.com/blog/2025/03/automatic-code-error-detection-agent/
57. Attacking Artificial Intelligence: AI's Security Vulnerability and What Policymakers Can Do About It | The Belfer Center for Science and International Affairs, accessed on July 14, 2025, https://www.belfercenter.org/publication/AttackingAI
58. What are the challenges of designing multi-agent systems? - Milvus, accessed on July 14, 2025, https://milvus.io/ai-quick-reference/what-are-the-challenges-of-designing-multiagent-systems
59. 9 Key Challenges in Monitoring Multi-Agent Systems at Scale - Galileo AI, accessed on July 14, 2025, https://galileo.ai/blog/challenges-monitoring-multi-agent-systems
60. The Confluence of Evolutionary Computation and Multi-Agent Systems: A Survey, accessed on July 14, 2025, https://www.ieee-jas.net/en/article/doi/10.1109/JAS.2025.125246
61. Formal Verification of Open Multi-Agent Systems - Imperial College ..., accessed on July 14, 2025, https://www.doc.ic.ac.uk/~pk3510/publications/AAMAS19-K+/paper.pdf

62. Formal Verification of Parameterised Neural-symbolic Multi-agent Systems - IJCAI, accessed on July 14, 2025, https://www.ijcai.org/proceedings/2024/12
63. Towards Formal Verification of Neuro-symbolic Multi-agent Systems - IJCAI, accessed on July 14, 2025, https://www.ijcai.org/proceedings/2023/800
64. Explainable AI: Transparent Decisions for AI Agents - Rapid Innovation, accessed on July 14, 2025, https://www.rapidinnovation.io/post/for-developers-implementing-explainable-ai-for-transparent-agent-decisions
65. Explainable and Transparent AI and Multi-Agent Systems | springerprofessional.de, accessed on July 14, 2025, https://www.springerprofessional.de/explainable-and-transparent-ai-and-multi-agent-systems/19368774
66. AI-Driven Scientific Research: The Revolution Has Begun | by noailabs | May, 2025, accessed on July 14, 2025, https://noailabs.medium.com/ai-driven-scientific-research-the-revolution-has-begun-ff95e285cc1c
67. Opinion: the future of science is automation | University of Cambridge, accessed on July 14, 2025, https://www.cam.ac.uk/research/news/opinion-the-future-of-science-is-automation
68. (PDF) Transformative Automation: AI in Scientific Literature Reviews - ResearchGate, accessed on July 14, 2025, https://www.researchgate.net/publication/378129477_Transformative_Automation_AI_in_Scientific_Literature_Reviews
69. Artificial intelligence to automate the systematic review of scientific literature - arXiv, accessed on July 14, 2025, https://arxiv.org/abs/2401.10917
70. AI Driven Autonomy- Future of in Business | Fusemachines Insights, accessed on July 14, 2025, https://insights.fusemachines.com/from-automation-to-autonomy-the-next-phase-of-ai-in-business/
71. The Next Generation of Consumer-Facing AI Personal Assistants: Empowering Everyday Life - Alcyone d.o.o., accessed on July 14, 2025, https://www.alcyone.si/index.php/en/blog/post/the-next-generation-of-consumer-facing-ai-personal-assistants-empowering-everyday-life
72. Towards Ethical Personal AI Applications: Practical Considerations for AI Assistants with Long-Term Memory - arXiv, accessed on July 14, 2025, https://arxiv.org/html/2409.11192v1
73. Ethics of Autonomous AI Agents: Risks, Challenges, Tips - Auxiliobits, accessed on July 14, 2025, https://www.auxiliobits.com/blog/the-ethics-of-autonomous-ai-agents-risks-challenges-and-tips/
74. The Ethical Challenges of AI Agents | Tepperspectives, accessed on July 14, 2025, https://tepperspectives.cmu.edu/all-articles/the-ethical-challenges-of-ai-agents/
75. New Ethics Risks Courtesy of AI Agents? Researchers Are on the Case - IBM, accessed on July 14, 2025, https://www.ibm.com/think/insights/ai-agent-ethics
76. The Rise of AI-Enabled Crime: Exploring the evolution, risks, and responses to AI-powered criminal enterprises - TRM Labs, accessed on July 14, 2025, https://www.trmlabs.com/resources/blog/the-rise-of-ai-enabled-crime-exploring-the-evolution-risks-and-responses-to-ai-powered-criminal-enterprises
77. Fully Autonomous AI Agents Should Not be Developed - arXiv, accessed on July 14, 2025, https://arxiv.org/html/2502.02649v2
78. 60+ Stats On AI Replacing Jobs (2025) - Exploding Topics, accessed on July 14, 2025, https://explodingtopics.com/blog/ai-replacing-jobs