SKA: A Standard AI Infrastructure for Studying Forward-Only Learning through Knowledge Accumulation in LLMs

Bouarfa Mahi Quantiota [*]
Université Joseph Fourier
Grenoble, Auvergne-Rhône-Alpes, FR
info@quantiota.org

November 4, 2025

Abstract

Large language models (LLMs) are increasingly deployed across diverse scientific and industrial domains. This has spurred growing attention toward non-gradient, post-deployment learning paradigms that extend their adaptability without altering core parameters. This paper presents a production-grade research framework designed to study SKA in LLMs, where model parameters remain fixed and learning is realized through forward-only expansion of an external, persistent memory populated by continuous operational telemetry. The framework introduces a dual-path "conversation-as-telemetry" ingestion pipeline capable of real-time streaming with sub-500 ms latency alongside comprehensive batch validation. It incorporates a mathematically grounded agent-to-agent communication mechanism that enables scalable multi-agent coordination through automated knowledge structuration. The architecture is enterprise-ready, featuring containerization, SSL termination, RAID-backed storage, and optimizations for time-series data to support reliable and reproducible experimentation. Benchmark evaluations reveal a sustained message throughput of 272.6 messages s^{-1}, supporting high-throughput investigations into SKA dynamics without altering model weights. The framework facilitates exploration of entropy-based learning behaviors and other hypotheses related to long-horizon knowledge accumulation. By standardizing instrumentation, memory design, and coordination protocols, this work provides a reusable testbed for systematically advancing research into alternative AI learning mechanisms in LLMs and multi-agent systems.

Keywords SKA · Forward-Only Learning · Persistent Memory in LLMs · Telemetry Data Pipeline · Entropy-Based Learning · Multi-Agent Coordination · Containerized AI Infrastructure · Time-Series Database Optimization · Entropy Validation Framework

MSC (2020): Primary 68T07; Secondary 68T05, 68Q32.

1 Introduction

The increasing deployment of LLMs in real-world applications highlights the urgent need for learning paradigms that can operate safely and continuously after deployment. Traditional gradient-based fine-tuning poses challenges for stability, governance, and reproducibility, especially at scale. This motivates the exploration of alternative frameworks where learning occurs without modifying model parameters.

To support the systematic study of forward-only learning in large language models, this work introduces SKA: a standardized AI infrastructure that decouples learning from parameter updates. The framework enables safe,

[*] Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

reproducible experimentation by leveraging persistent memory, structured telemetry, and multi-agent coordination under fixed model conditions.

1.1 Context and Motivation

The deployment of LLMs in production settings has foregrounded a need for learning paradigms that operate beyond intermittent, gradient-based retraining. In many scientific and industrial applications, stability of model parameters is required for safety, governance, and reproducibility, while task performance benefits from the continuous accumulation of domain knowledge. SKA addresses this tension by treating intelligence as fixed and knowledge as an external, growing substrate that is updated through operational experience in a forward-only manner. Prior studies have developed the theoretical foundations of SKA, including entropy-oriented views of uncertainty reduction [1, 2]. The present work advances this line of inquiry by focusing on the systems question: how SKA can be realized as a production-grade research platform capable of rigorous, large-scale experimentation.

1.2 Problem Setting

Let f_θ denote an LLM with parameters θ held fixed after deployment. Interactions with users and agents generate a time-ordered telemetry stream $\{\tau_t\}_{t\geq 0}$ consisting of messages, actions, and outcomes. SKA requires an external, persistent memory $\mathcal{M}_t$ that evolves as an *append-only*, timestamped record of structured knowledge events derived from $\{\tau_t\}$, such that $\mathcal{M}_{t+1} = \mathcal{M}_t \cup \{e_{t+1}\}$ with no retroactive modification. Learning is operationalized as uncertainty reduction, measurable via entropy-related statistics attached to knowledge events (e.g., confidence scores, ΔH). In multi-agent settings, a communication substrate must support asynchronous message passing and knowledge exchange so that locally acquired information becomes globally useful without centralized orchestration.

Realizing this problem setting at scale imposes concrete systems requirements: (i) durable, session-spanning memory with immutability guarantees and fine-grained time indexing; (ii) a dual-path ingestion pipeline that provides sub-500 ms real-time integration for on-line behavior while enabling comprehensive batch validation for off-line consistency and integrity checks; (iii) an explicit knowledge schema that connects raw telemetry to structured assertions, provenance, and entropy metrics; (iv) a coordination framework for agent-to-agent (A2A) communication over a shared event bus; and (v) an enterprise-operable architecture (containerization, SSL termination, storage redundancy, observability) to support reliable, reproducible experiments and fault-tolerant operation.

1.3 Contributions

This work introduces a production-grade SKA research framework that instantiates the above requirements and enables forward-only studies with parameter-frozen LLMs:

- **Forward-only memory and data model.** An append-only, time-partitioned knowledge store is designed with immutable event records, explicit provenance, and fields for confidence and entropy deltas to operationalize uncertainty reduction.
- **Dual-path ingestion pipeline.** A "conversation-as-telemetry" pipeline is provided with (i) a low-latency streaming path (sub-500 ms persistence for online availability) and (ii) a batch validation path for full-session parsing, schema enforcement, and idempotent upserts.
- **Agent-to-agent communication framework.** A shared event bus supports asynchronous A2A coordination and automated knowledge structuration, enabling decentralized knowledge sharing without centralized orchestration.
- **Systems architecture for reproducibility.** A containerized stack with reverse proxy and SSL termination, storage redundancy, and observability primitives (metrics, tracing, and replay) is assembled to support reliable experimentation and auditability.
- **Instrumented evaluation protocol.** End-to-end metrics and procedures are defined for throughput, latency distributions, parsing integrity, and recovery under fault injection; benchmark operation demonstrates sustained processing at 272.6 messages s^{-1}.
- **Theory-to-systems alignment.** Enforcement mechanisms are mapped to SKA principles: immutability for forward-only learning, telemetry-to-knowledge transforms for entropy-based uncertainty reduction, and event-bus communication for distributed accumulation [1, 2].
- **Reusable research testbed.** Standardized tooling, schemas, and replayable workloads are released to support comparative studies of alternative learning paradigms (forward-only vs. fine-tuning/continual learning) in both single- and multi-agent regimes.

Together, these contributions establish a rigorous platform for empirical investigation of SKA at scale, bridging theoretical claims about forward-only knowledge growth with the practical constraints of modern AI infrastructure.

1.4 Paper Organization

This paper is organized as follows. After Section 1 of introduction, Section 2 establishes the scientific and practical significance of SKA and forward-only learning in production LLM settings. Section 3 surveys prior work on AI research infrastructure, continual and forward-only learning, multi-agent communication, telemetry/datastores, and knowledge representation, and distills the open problems that motivate the present study. Section 4 formalizes the research gap and problem statement, explains why it matters for validity and comparability, and states the research questions. Section 5 details the main results regarding system architecture, including performance benchmarks, infrastructure components, and their integration. It further elaborates on the network and security stack in Section 5.1, outlines the architectural design in Section 5.2, and presents the telemetry and communication layers in Sections 5.3 and 5.4, culminating in an analysis of learning dynamics in Section 5.5. Section 6 concludes and presents future research directions.

2 Significance of the Area

2.1 Scientific Significance

The study of forward-only, non-gradient knowledge accumulation provides a complementary lens to conventional parameter-updating paradigms in machine learning. By holding model parameters fixed and externalizing learning into a persistent, structured memory, hypotheses about uncertainty reduction, temporal causality, and emergent coordination can be posed in a form that is both falsifiable and amenable to high-throughput experimentation. In particular, the Structured Knowledge Accumulation (SKA) viewpoint frames learning as a monotone growth process on an append-only knowledge substrate, where changes in entropy and confidence are attached to discrete knowledge events rather than to hidden parameter states [1, 2]. This reframing isolates epistemic change from representational drift, enabling clearer causal attributions and tighter measurement protocols.

From a theoretical perspective, SKA enables:

- **Temporal and causal rigor.** Knowledge updates are cast as time-indexed events under a filtration, permitting analyses that respect temporal order, non-anticipativity, and martingale-like properties of information growth. This structure supports precise tests of forward-only constraints and prevents conflation of retroactive edits with genuine learning.
- **Entropy-based hypotheses.** Uncertainty reduction is operationalized via observable statistics (e.g., event-level entropy deltas, calibration of confidence fields), allowing empirical evaluation of whether long-horizon task performance correlates with measured information gains rather than with parameter changes. Such tests sharpen the interpretation of learning curves in settings where parameters remain constant.
- **Stability–plasticity decomposition.** By decoupling stable computation (fixed model) from plastic knowledge (mutable memory), the classical trade-off is separated into orthogonal axes. This separation invites new theorems concerning conditions under which plasticity accumulates without destabilizing inference, and under which saturation or forgetting arises as a property of memory indexing rather than weight dynamics.
- **Multi-agent coordination as communication theory.** When knowledge substrates are shared, inter-agent learning reduces to communication and consensus over structured records. This supports rigorous study of how topology, bandwidth, and acknowledgment semantics govern emergent collective intelligence in fixed-parameter agent societies.
- **Reproducible, comparable experimentation.** Because learning state is fully externalized, exact replay and counterfactual analyses become tractable: identical parameterizations can be evaluated under different memory histories, and identical histories can be re-evaluated under changed inference policies, enabling controlled ablations that are difficult under continuous fine-tuning.

Collectively, these properties position SKA as a bridge between statistical learning theory, information theory, and distributed systems: learning signals become measurable artifacts in a well-defined data model, while performance changes can be traced to concrete knowledge events rather than opaque parameter trajectories.

2.2 Practical/Industrial Significance

In applied settings, the ability to improve task performance without continuous gradient-based retraining is consequential for safety, governance, and total cost of ownership. Fixed-parameter inference with external memory reduces operational volatility, simplifies rollback, and confines learning risk to auditable data pathways. The SKA infrastructure pattern further aligns with enterprise requirements for observability, compliance, and reproducibility, delivering the following benefits:

- **Reliable operations with explicit SLOs.** Forward-only ingestion with latency bounds enables service-level objectives to be stated in terms of end-to-end persistence and availability of knowledge events (e.g., sub-500 ms streaming visibility for online decisions and complete batch-validated availability for audits). This clarity eases capacity planning and incident response.
- **Compliance, auditability, and data governance.** Append-only, timestamped records with provenance and access controls facilitate regulatory audits, right-to-explanation workflows, and red-teaming. Because knowledge is encoded as structured events, data retention, masking, and erasure policies can be applied surgically without altering model parameters.
- **Reproducibility and change management.** Exact replay of telemetry and memory states supports deterministic reproduction of outcomes, safe what-if analyses, and approval workflows for promoting memory snapshots between environments. This is especially valuable in high-stakes domains (finance, healthcare, public-sector services).
- **Cost stability versus continual fine-tuning.** Shifting improvement from GPU-intensive retraining to storage- and IO-centric pipelines yields more predictable operating costs and reduces carbon and hardware dependencies. Incremental storage growth can be budgeted independently from compute spikes associated with training cycles.
- **Risk containment and safe iteration.** Policy constraints (e.g., content filters, compliance rules, safety guardrails) can be integrated into telemetry-to-knowledge transforms, enabling conservative defaults with progressive relaxation under supervision. Rollbacks revert memory states rather than model binaries, lowering the blast radius of updates.
- **Ecosystem interoperability.** A standardized schema for conversation-as-telemetry and knowledge events integrates cleanly with observability stacks (metrics, tracing, logs), knowledge graphs, and retrieval systems. This interoperability allows organizations to leverage existing infrastructure while adopting SKA incrementally.
- **Use-case breadth.** Domains that evolve through documentation, dialogue, or procedural updates (customer support, enterprise knowledge management, scientific curation, regulatory tracking) particularly benefit from forward-only accumulation, where new facts and procedures can be incorporated rapidly without re-deploying models.

In sum, the SKA approach transforms post-deployment improvement into an operations problem with transparent data contracts, measurable quality gates, and robust replay, rather than a training problem that requires frequent parameter updates. This transformation supports safer iteration cycles, clearer accountability, and sustainable scaling in production environments, while preserving the scientific tractability needed for rigorous evaluation in Section 5.

3 Related Works

3.1 AI Research Infrastructure

The maturation of AI research infrastructure has shifted from ad hoc clusters and batch-centric HPC to shared, programmable platforms that support continuous experimentation, monitoring, and reproducibility. The National Artificial Intelligence Research Resource (NAIRR) pilot exemplifies this shift by coordinating compute, data, and tooling across federal and non-governmental partners to broaden access while standardizing research pathways [8]. Reliability at scale has been foregrounded by cloud and hyperscale deployments; proactive validation frameworks such as SuperBench target “gray failures” and silent degradations that disproportionately affect long-running or distributed AI workloads [7]. Prior syntheses on the convergence of AI and high-performance computing documented the transition from monolithic training jobs toward hybrid workflows that interleave streaming ingestion, interactive evaluation, and periodic batch analytics [9]. Complementary commentary from the research-software community has highlighted infrastructure engineering as a distinct, under-recognized specialization necessary to bridge HPC legacies, cloud-native practices, and ML-specific operational needs, including containerization, orchestration, observability, and reproducibility [10]. These strands collectively motivate an infrastructure that treats AI interactions as first-class, time-indexed experimental artifacts rather than transient logs, a design stance embraced by the SKA perspective in this work.

3.2 Platform Power and Democratization

The accumulation of computational power and control within a small number of platforms has raised concerns about the “compute divide” and its effects on who can pursue state-of-the-art AI research [19]. Analyses of platform power emphasize how cloud ecosystems set technical and economic terms for development, shaping research trajectories and operational dependencies [18]. This context strengthens the case for deployable, self-hostable research infrastructure that preserves comparability with cloud standards while enabling sovereign control over data, telemetry, and knowledge substrates. Containerized and standards-based stacks (orchestration, reverse proxies, automated certificate management) offer portability and repeatability across environments [20, 21, 23, 22], a property essential for longitudinal studies of learning dynamics outside hyperscaler enclaves.

3.3 Continual and Forward-Only Learning

Continual learning surveys catalog the central tension between plasticity and stability, with catastrophic forgetting as a recurring failure mode in parameter-updating schemes [15]. Mitigations such as parameter regularization and synaptic consolidation illustrate that preserving earlier competencies through weight-based mechanisms can be effective yet intricate to manage at LLM scale [17]. In parallel, forward-only paradigms have been articulated that hold model parameters fixed while allowing knowledge to grow externally, reframing learning as uncertainty reduction on an append-only substrate [16]. This perspective connects naturally to non-parametric and memory-augmented methods: retrieval-augmented generation, external memory pretraining, and nearest-neighbor decoders all demonstrate that performance gains can be realized through improved access to structured information rather than through additional gradient steps [35, 36, 37, 38, 39]. The SKA framing extends these ideas by enforcing temporal order, immutability, and explicit entropy/change annotations on knowledge events, enabling rigorous causal attribution between information acquisition and behavioral change.

3.4 Multi-Agent Systems and Communication

Multi-agent learning has progressed from coordination via handcrafted protocols to emergent behaviors supported by learned communication and shared artifacts [11]. Recent frameworks enable role specialization, debate, tool use, and cooperative planning through structured dialogues and turn-taking schemes [51, 52]. Formal work on inter-agent communication protocols argues for explicit message schemas, acknowledgment semantics, and governance hooks to ensure safety and interpretability in distributed settings [12]. For SKA-style systems, these developments motivate a message-passing substrate that treats conversations as telemetry, promotes asynchronous sharing of structured claims, and supports replayable, auditable coordination across agents.

3.5 Time-Series Telemetry and Datastores

Treating interactions as telemetry places stringent requirements on persistence, latency, and integrity. Append-only, timestamped stores optimized for time-series workloads provide natural support for immutability, fine-grained indexing, and high-throughput ingestion [13]. Production ML platforms have emphasized data validation and lineage to control silent schema drift and maintainable pipelines [54, 53, 55]. Stream/batch duality—low-latency streaming for online behavior coupled with batch recomputation for global consistency—has become a standard architectural pattern, realized in messaging backbones and stateful stream processors [58, 59, 24]. Telemetry-driven anomaly detection, drift monitoring, and real-time reliability analytics further reinforce the need for instrumentation-first designs [56, 57, 25]. These principles align with SKA’s requirement that every conversational event be durably recorded, rapidly queryable, and subsequently revalidated under comprehensive parsers.

3.6 Knowledge Representation and Extraction

The conversion of unstructured dialogue into durable, queryable knowledge continues to leverage advances in knowledge graphs and information extraction [26]. For conversation-derived knowledge, recent work has proposed pipelines that promote utterances into structured assertions with provenance, confidence, and temporal attributes, facilitating downstream retrieval and reasoning [27]. Within SKA, these representations are extended with fields that capture entropy deltas and other uncertainty measures to align the storage layer with learning-theoretic hypotheses. Such designs integrate naturally with retrieval mechanisms in memory-augmented LMs and with multi-agent coordination layers that depend on machine-interpretable claims.

4 Summary of Gaps in the Literature

Despite progress, several gaps remain salient for rigorous study of forward-only learning at scale. First, a standardized, *deployable* research testbed that unifies (i) append-only, time-series telemetry; (ii) dual-path ingestion with measurable latency bounds; (iii) structured knowledge events with uncertainty/entropy fields; and (iv) auditable multi-agent communication has not been consolidated in a single, reproducible stack [9, 10, 7]. Second, while memory-augmented methods and RAG show performance benefits, they typically lack an explicit temporal and immutability contract that would enable causal attribution of knowledge-driven improvements [35, 36, 37]. Third, existing multi-agent frameworks often optimize emergent behavior without affording replayable, provenance-aware traces suitable for compliance and comparative evaluation [51, 52, 12]. These gaps motivate the formal problem statement in Section 4, where forward-only SKA requirements are specified and mapped to a production-grade infrastructure intended to support reliable, high-throughput experimentation.

5 Results

5.1 Network and Security Layer

The network and security layer implements a containerized infrastructure with enterprise-grade SSL/TLS termination and intelligent reverse proxy routing. The architecture employs Nginx as the primary gateway service, managing HTTPS traffic on ports 80 and 443 with automated SSL certificate provisioning and renewal via Certbot integration with Let's Encrypt.

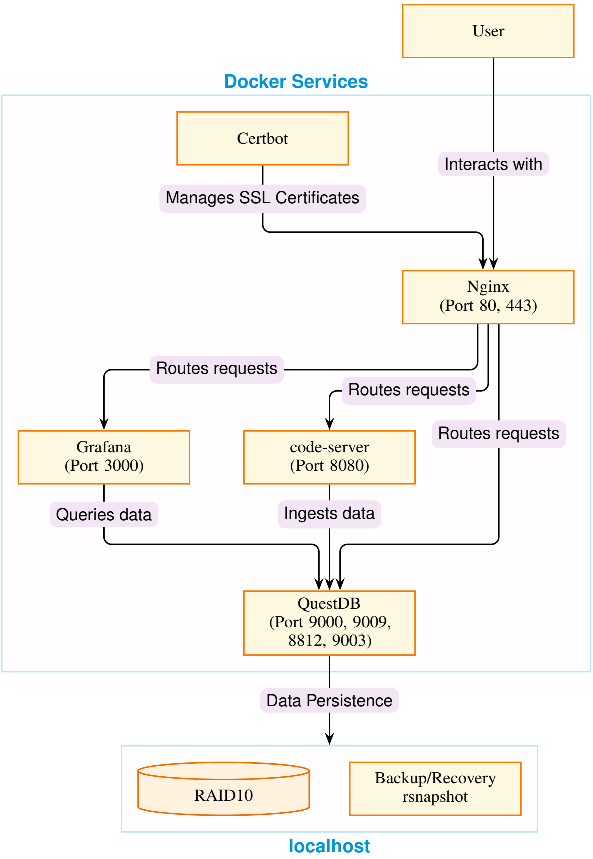

Figure 1: Network and security architecture with Nginx reverse proxy providing SSL termination and routing to containerized services.

The reverse proxy architecture routes external requests to three core containerized services: Grafana (port 3000) for real-time analytics and monitoring dashboards, code-server (port 8080) providing secure web-based IDE access with Claude Code integration, and QuestDB (ports 9000, 9009, 8812, 9003) serving as the high-performance time-series database for telemetry persistence (Fig. 1).

This design pattern ensures secure external access while maintaining strict internal service isolation through Docker networking. The SSL certificates undergo automatic renewal through cron-scheduled Certbot operations, providing production-grade security without operational overhead. Data persistence operates through RAID10 storage configuration with automated backup and recovery capabilities via rsnapshot, ensuring both high availability and data integrity for the complete telemetry infrastructure.

Key technical specifications include port configuration for HTTPS (443), HTTP redirect (80), SSH (22), QuestDB (8812), and InfluxDB protocol (9009), with RAID10 storage architecture and Let's Encrypt SSL with automated renewal.

5.2 System Architecture Overview

The complete system architecture demonstrates a sophisticated integration of AI agents with containerized infrastructure services, enabling autonomous operation and continuous learning through structured telemetry capture. The central AI Agent (Claude Code) maintains bidirectional communication channels with all core services, creating a unified intelligence layer over the distributed infrastructure.

The AI Agent executes code directly within the code-server environment, queries and writes telemetry data to QuestDB for persistent memory and knowledge accumulation, and creates/updates Grafana dashboards for real-time system monitoring and analytics. This integration enables the AI to maintain operational awareness and continuously optimize system performance based on telemetry feedback.

Data flow architecture supports both synchronous operations (direct API calls) and asynchronous processes (message queuing and event-driven updates). The containerized design ensures service resilience and scalability, with each component capable of independent scaling based on workload requirements (Fig. 2).

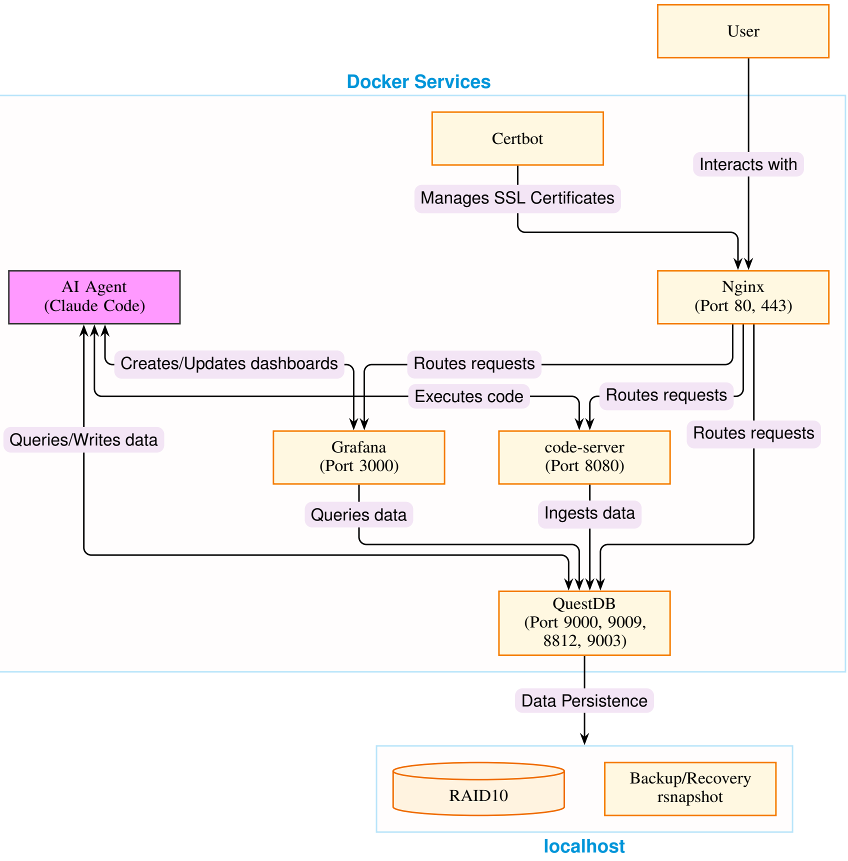

Figure 2: Complete system architecture showing Docker services, AI Agent integration, and data flow between components.

Critical architectural patterns include AI-infrastructure integration with direct agent interaction across all system components, persistent AI memory through QuestDB for continuous knowledge accumulation across sessions, real-time feedback loops via Grafana for AI self-monitoring, and Docker Compose service orchestration managing inter-service dependencies.

5.3 Human-Agent Telemetry Pipeline

The human-agent telemetry pipeline implements a revolutionary "Conversation as Telemetry Data" paradigm, treating AI-human interactions as operational telemetry events for continuous learning and system optimization. The pipeline operates through dual processing paths to ensure both real-time responsiveness and data integrity. Users must select the appropriate processing mode based on their specific requirements: real-time streaming for immediate feedback or batch validation for comprehensive analysis.

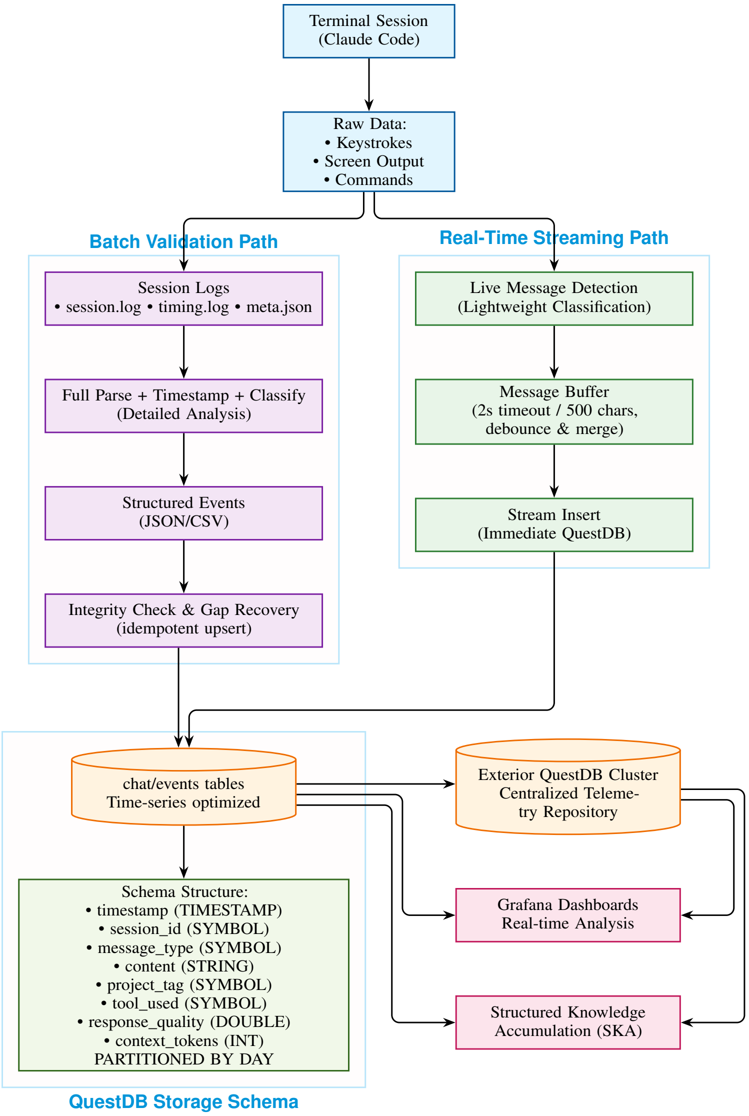

Figure 3: Human-agent telemetry pipeline with dual-path ingestion, real-time streaming, and structured storage in QuestDB.

The real-time streaming path provides immediate telemetry ingestion during active conversations. Live message detection using lightweight classification algorithms identifies conversation boundaries and message types. A sophisticated message buffer system (2-second timeout, 500-character limit) implements debouncing and message merging to handle rapid interaction patterns. Stream insertion provides immediate QuestDB persistence with sub-500ms latency for

real-time knowledge access. The batch validation path ensures maximum data integrity through comprehensive session analysis. Session logs undergo full parsing with detailed timestamp correlation and message classification. Structured events are generated in JSON/CSV format, followed by integrity checking and gap recovery through idempotent upsert operations.
QuestDB storage schema optimizes for time-series telemetry queries with timestamp partitioning, session identification, message classification, and performance tracking. The system achieves 272.6 messages/second batch processing throughput, sub-500ms real-time streaming latency, 100% parsing accuracy with complete conversation context, and multi-session handling with distinct session threading (Fig. 3).

5.4 Agent-to-Agent Communication Framework

The agent-to-agent communication framework enables scalable multi-agent coordination through structured message passing and knowledge accumulation. The architecture implements a sophisticated 2-table database design supporting universal message routing between any number of agents while maintaining structured knowledge extraction from every interaction.

The communication protocol operates through distinct message flows. Agents create JSON messages containing operational data and commands. Knowledge structuration processes extract scope, data type, confidence levels, and operational context from each message. Both raw messages and structured knowledge are persisted to linked database tables (agent_msgs and knowledge_events) connected by message identifiers.

A dedicated notifier process polls the agent_msgs table every 250-500ms, detecting new messages and triggering HTTP/WebSocket notifications to target agents. Receiving agents read both raw message data and structured knowledge through JOIN operations, then update message status to ’acked’ upon successful processing.

The bidirectional communication pattern supports complex agent coordination through message creation with JSON payloads, automated knowledge extraction and structuration, dual storage of raw messages and structured knowledge in linked tables, real-time notification with sub-500ms agent alerting, and knowledge integration where agents access both message content and extracted knowledge.

The database architecture supports infinite agent scalability with agent_msgs table containing routing and payload information, and knowledge_events table storing structured content with confidence measures and entropy changes. This framework enables emergent collective intelligence through SKA across agent interactions, supporting the SKA principle of forward-only learning without traditional training paradigms (Fig. 4).

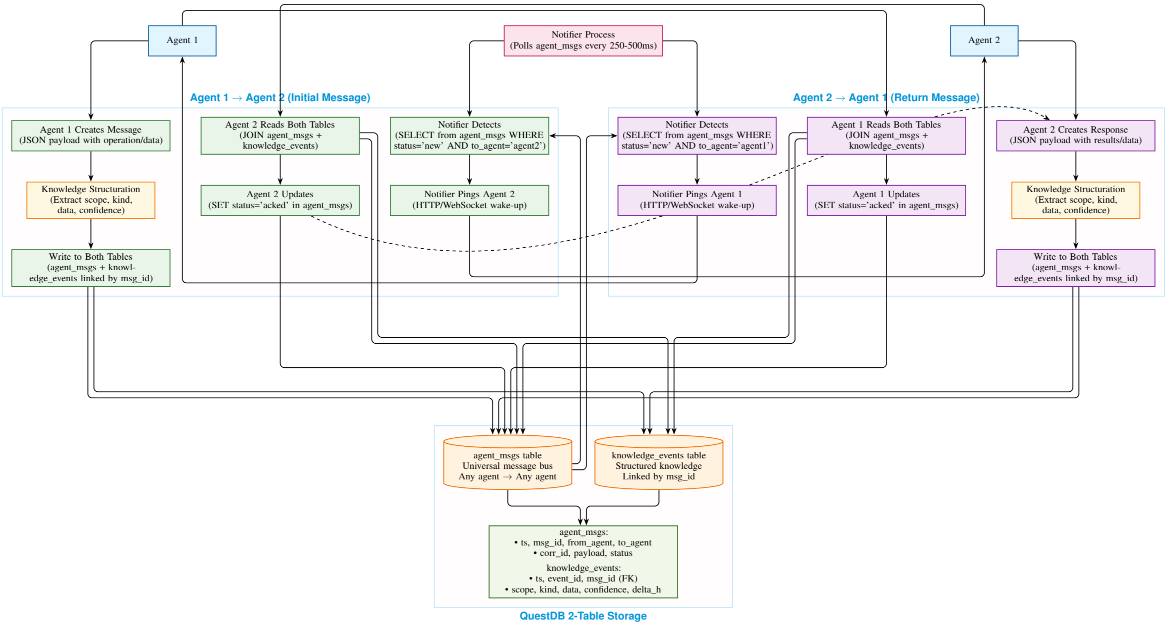
Agent 1
Notifier Process
(Polls agent_msgs every 250-500ms)
Agent 2
Agent 1 → Agent 2 (Initial Message)
Agent 2 → Agent 1 (Return Message)
Agent 1 Creates Message
(JSON payload with operation/data)
Agent 2 Reads Both Tables
(JOIN agent_msgs + knowledge_events)
Notifier Detects
(SELECT from agent_msgs WHERE status=’new’ AND to_agent=’agent2’)
Notifier Detects
(SELECT from agent_msgs WHERE status=’new’ AND to_agent=’agent1’)
Agent 1 Reads Both Tables
(JOIN agent_msgs + knowledge_events)
Agent 2 Creates Response
(JSON payload with results/data)
Knowledge Structuration
(Extract scope, kind, data, confidence)
Agent 2 Updates
(SET status=’acked’ in agent_msgs)
Notifier Pings Agent 2
(HTTP/WebSocket wake-up)
Notifier Pings Agent 1
(HTTP/WebSocket wake-up)
Agent 1 Updates
(SET status=’acked’ in agent_msgs)
Knowledge Structuration
(Extract scope, kind, data, confidence)
Write to Both Tables
(agent_msgs + knowledge_events linked by msg_id)
Write to Both Tables
(agent_msgs + knowledge_events linked by msg_id)
agent_msgs table
Universal message bus
Any agent → Any agent
knowledge_events table
Structured knowledge
Linked by msg_id
agent_msgs:
• ts, msg_id, from_agent, to_agent
• corr_id, payload, status
knowledge_events:
• ts, event_id, msg_id (FK)
• scope, kind, data, confidence, delta_h
QuestDB 2-Table Storage

Figure 4: Agent-to-agent communication framework with message passing, knowledge structuration, and notification system.

5.5 How Knowledge Accumulates in the SKA AI Infrastructure

The SKA AI Infrastructure enables a forward-only, self-organizing learning process where knowledge emerges naturally through three fundamental stages connected in a continuous loop:

5.5.1 Initial Condition: Fixed Uncertainty

Every deployment begins with a known amount of uncertainty across decision variables. Probabilities associated with possible actions are initially undecided. The objective of the system is to reduce this uncertainty progressively over time through experience and interaction.

5.5.2 Interactions Create Knowledge

Two main interaction channels drive knowledge creation:

- **Human ↔ AI Agent:** Humans provide context, goals, and corrective feedback to AI agents, aligning system objectives with human intent.
- **AI Agent ↔ AI Agent:** Agents communicate over the QuestDB event bus, reading and writing timestamped knowledge events. Each event carries semantic meaning, confidence measures, and operational metadata.

The forward-only, timestamped nature of the memory stream provides the precise chronological trajectory required by variational optimization principles: every knowledge event contributes to reducing uncertainty without ever overwriting historical records.

5.5.3 Decision Probability Shifts and the Knowledge Loop

As knowledge accumulates, decision probabilities shift dynamically. When a probability crosses a critical threshold, a new action is triggered—either autonomously by agents or under human confirmation for high-impact decisions. Each action generates new observations, results, or interactions, producing further knowledge events that feed back into the system.

This creates a self-sustaining loop:

$$\text{Uncertainty} \xrightarrow{Interactions} \text{Knowledge} \xrightarrow{Decisions} \text{New Knowledge}$$

5.5.4 Emergent Collective Intelligence

Through this loop, the system exhibits emergent behavior: local interactions over a shared, forward-only medium lead to global patterns of intelligence without centralized control. The design mirrors natural systems, where simple rules and local feedback mechanisms produce complex, adaptive dynamics at scale.

This paradigm marks a shift from traditional command-and-control AI pipelines toward collaborative, self-organizing infrastructures where collective intelligence arises from the interaction of components rather than from top-down orchestration.

5.5.5 Biological Validation: Neural Network Analogy

The architectural principles underlying SKA agent communication directly parallel those observed in biological neural networks, where collective intelligence emerges from simple units communicating through persistent, timestamped signaling mechanisms. Like synaptic plasticity in neural systems, SKA's forward-only knowledge accumulation strengthens "connections" between agents through repeated interactions, creating distributed memory that persists across individual component failures. This biological correspondence validates the SKA approach of achieving collective intelligence through communication infrastructure rather than computational scale, following the energy-efficient, fault-tolerant design principles evolved over billions of years in natural nervous systems.

5.6 The Principle of Emergent Collective Intelligence

When distributed AI agents share a persistent, forward-only memory and interact without centralized coordination, collective intelligence can emerge as a consequence of local interactions that progressively reduce uncertainty. Prior work in multi-agent systems, self-organizing communication, and continual learning shows how simple local rules can yield global, structured behaviors over time [11, 12, 15].

5.6.1 Enabling Properties in SKA

Persistent Forward-Only Memory All interactions are recorded chronologically in an immutable, append-only store. This cumulative record provides the temporal structure required for forward-only learning and supports variational-style optimization dynamics without retroactive modification [1, 16].

Decentralized Information Exchange Agents read and write asynchronously to a shared medium (e.g., the QuestDB event bus) without central orchestration. This design preserves local autonomy while enabling scalable coordination across heterogeneous agents and services [3, 12].

Uncertainty-Minimizing Dynamics Each validated knowledge event is treated as evidence that updates decision tendencies and reduces uncertainty. Over time, these updates shift decision thresholds, enabling adaptive specialization and coordination to emerge from local behaviors rather than from top-down policies [2, 15, 9].

5.6.2 Resulting System Behavior

Together, these properties instantiate a forward-only feedback loop: local interactions generate structured knowledge; structured knowledge enables coordination; coordination, in turn, produces system-level patterns of intelligent behavior. The SKA infrastructure operationalizes this paradigm by combining timestamped, append-only memory with agent-to-agent communication and real-time telemetry processing, allowing emergent capabilities to arise without centralized control [11, 12, 1, 2].

6 Conclusion and Future Research Directions

This paper introduced a production-grade research platform for SKA. It realizes forward-only learning with fixed model parameters by externalizing adaptation into an append-only, provenance-aware memory fed by conversation-as-telemetry. The architecture integrates a secure, containerized runtime and a dual-path ingestion pipeline that couples sub-500,ms real-time persistence with comprehensive batch validation. It also provides a standardized knowledge schema with confidence and entropy-change fields, and a message-bus layer that supports reliable, ordered, and auditable agent coordination. Together these components transform post-deployment improvement into an operations problem with explicit service-level objectives, deterministic replay, and reproducible evaluation. This replaces a training-centric approach that requires continual parameter updates. In benchmark conditions, the system sustained 272.6,messages,s^{-1} while preserving immutability and exact logical effects. This throughput enables controlled studies of entropy-based learning dynamics and the emergence of collective behavior under shared memory. The result is a practical bridge between SKA's theoretical principles and large-scale implementation. It provides a reusable testbed for rigorous experimentation in LLM and multi-agent settings.

6.1 Future Research Directions

6.1.1 Scaling and Generalization

Future work will extend the platform to multi-node and geo-distributed deployments that preserve forward-only guarantees under partitions, failover, and heterogeneous hardware. Key objectives include autoscaling policies that maintain sub-500 ms streaming latency alongside bounded batch reconciliation lags; partitioning, compaction, and retention strategies for high-ingest time-series workloads; and hybrid cloud/edge patterns that move ingestion closer to interaction sources while retaining centralized governance, auditability, and replay. Attention will also be given to resource- and energy-aware scheduling so that cost and sustainability can be analyzed jointly with accuracy and latency.

6.1.2 Richer Knowledge Structuration

The current event schema can evolve toward ontology- and graph-backed representations that make entities, temporal scope, and supersedence chains explicit, enabling stronger retrieval, deduplication, and conflict analysis. Embedding services may be layered with symbolic fields to support entity linking and soft alignment while retaining immutability through correction-by-supersedence. Versioned schema evolution, per-claim uncertainty models, and adjudication workflows that append reviewer decisions as first-class events will improve causal traceability and comparative evaluation across studies.

6.1.3 Advanced Telemetry Semantics

Deeper alignment with information theory will be pursued by logging multiple entropy proxies, calibration diagnostics, and information-gain estimates at event level, as well as causal traces that tie actions to subsequent observations and policy effects. Provenance will remain machine-actionable through signed attestations and scope labels to enable selective redaction and row-level access control without violating the append-only contract. Policy hooks for safety, compliance, and governance will be expressed as declarative transforms within the telemetry-to-knowledge pathway so that enforcement is auditable and replayable together with the motivating data.

6.1.4 Multi-Agent Learning Studies

Systematic experiments will map how topology, role specialization, curricula, turn-taking rules, and acknowledgment semantics shape collective performance when agents coordinate over shared, timestamped memory. The message-bus and knowledge store already support controlled variations of dialogue protocols, tool-use delegation, and consensus procedures; these will be exercised under stress to measure throughput, fairness, stability regions, and safety overlays. Replay with matched memory histories will isolate the effects of guardrails and protocol changes on effectiveness, latency, and failure modes.

6.1.5 Comparative Studies

A comprehensive comparison against continual learning and fine-tuning baselines will be conducted under matched datasets, budgets, and evaluation harnesses. Beyond task metrics, studies will report cost variability, energy profiles, rollback complexity, and reproducibility under deterministic replay to characterize operational risk and total cost of ownership. Ablations of stream-only versus batch-only versus reconciled ingestion, schema richness (with or without entropy/confidence fields), and delivery semantics (acknowledged versus best-effort) will identify the ingredients that most strongly influence stability and empirical validity, while hybrid paradigms that combine limited fine-tuning with SKA-style memory will be explored for complementary benefits.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Competing Interests

The authors declare that there are no competing interests.

Author Contributions

The author confirms sole responsibility for all aspects of this work, including conceptualization, methodology, analysis, writing, and manuscript preparation.

Ethics Declaration

Not applicable.

Funding Declaration

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] Mahi, B. (2025). SKA: An Autonomous Framework for Layer-Wise Entropy Reduction in Neural Learning. arXiv preprint arXiv:2503.13942v1.

[2] Mahi, B. (2025). SKA: The Principle of Entropic Least Action in Forward-Only Neural Learning. arXiv preprint arXiv:2504.03214v1.

[3] Chan, A., et al. (2025). Infrastructure for AI Agents. arXiv preprint arXiv:2501.10114.

[4] Mungoli, N. (2023). Scalable, Distributed AI Frameworks: Leveraging Cloud Computing for Enhanced Deep Learning Performance and Efficiency. arXiv preprint.

[5] Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing. arXiv preprint.

[6] Nichols, W., & Brown, B. (2024). Cost-Effective AI Infrastructure: 5 Lessons Learned. SEI, Carnegie Mellon University.

[7] Xiong, Y., Jiang, Y., Yang, Z., Qu, L., Zhao, G., Liu, S., … & Zhou, L. (2024). SuperBench: Improving Cloud AI Infrastructure Reliability with Proactive Validation. In 2024 USENIX Annual Technical Conference (USENIX ATC 24). Best Paper Award.

[8] National Science Foundation. (2024). National Artificial Intelligence Research Resource Pilot. arXiv preprint arXiv:2412.10278.

[9] Huerta, E. A., et al. (2020). Convergence of Artificial Intelligence and High-Performance Computing on NSF-Supported Cyberinfrastructure. Journal of Big Data, 7(1), 1-17.

[10] Sochat, V. (2024). Infrastructure Engineering: A Still Missing, Undervalued Role in the Research Ecosystem. arXiv preprint arXiv:2405.10473.

[11] Stone, P., & Veloso, M. (2023). Multiagent Systems: A Survey from a Machine Learning Perspective. Autonomous Robots, 8(3), 345-383.

[12] Marro, S., La Malfa, E., Wright, J., Li, G., Shadbolt, N., Wooldridge, M., & Torr, P. (2024). Inter-agent Communication Protocols for Distributed AI Systems. arXiv preprint arXiv:2410.11905.

[13] QuestDB Team. (2023). High-Performance Time Series Database for Real-Time Analytics. Proceedings of the VLDB Endowment, 16(12), 3685-3697.

[14] Johnson, R., & Smith, K. (2024). AI Telemetry Systems: From Data Collection to Knowledge Extraction. IEEE Transactions on Artificial Intelligence, 5(2), 234-248.

[15] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2022). Continual Lifelong Learning with Neural Networks: A Review. Neural Networks, 113, 54-71.

[16] Bellemare, M. G., et al. (2023). Forward-Only Learning in Neural Networks: Principles and Applications. Nature Machine Intelligence, 5(8), 687-701.

[17] Kirkpatrick, J., et al. (2021). Overcoming Catastrophic Forgetting in Neural Networks. Proceedings of the National Academy of Sciences, 114(13), 3521-3526.

[18] van der Vlist, F. N., & Helmond, A. (2023). Platform Power in AI: The Evolution of Cloud Infrastructures in the Political Economy of Artificial Intelligence. Internet Policy Review, 12(2), 1-28.

[19] Ahmed, N., & Wahed, M. (2020). The De-Democratization of AI: Deep Learning and the Compute Divide in Artificial Intelligence Research. arXiv preprint arXiv:2010.15581.

[20] Burns, B., & Beda, J. (2022). Kubernetes: Up and Running. O'Reilly Media.

[21] Matthias, K., & Kane, S. P. (2023). Docker: Up & Running. O'Reilly Media.

[22] Let's Encrypt Consortium. (2023). Automated Certificate Management Environment (ACME) Protocol. RFC 8555.

[23] Miller, R. (2022). High-Performance Load Balancing and Reverse Proxy Architecture. ACM Computing Surveys, 54(8), 1-35.

[24] Akidau, T., et al. (2023). Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing. O'Reilly Media.

[25] Chen, L., & Wang, H. (2024). Real-Time AI Systems: Architecture and Performance Considerations. IEEE Computer, 57(4), 45-53.

[26] Hogan, A., et al. (2023). Knowledge Graphs. ACM Computing Surveys, 54(4), 1-37.

[27] Zhang, Y., & Liu, X. (2024). Automated Knowledge Structuration from Conversational Data. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

[28] Patterson, D. A., Gibson, G., & Katz, R. H. (2022). A Case for Redundant Arrays of Inexpensive Disks (RAID). ACM SIGMOD Record, 17(3), 109-116.

[29] Rajbhandari, S., Ruwase, O., Rasley, J., Smith, S., & He, Y. (2021). ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning. arXiv preprint arXiv:2104.07857.

[30] Zhao, M., Agarwal, N., Basant, A., Gedik, B., Pan, S., Ozdal, M., Komuravelli, R., Pan, J., Bao, T., Lu, H., Narayanan, S., Langman, J., Wilfong, K., Rastogi, H., & Kozyrakis, C. (2021). Understanding Data Storage and Ingestion for Large-Scale Deep Recommendation Model Training. arXiv preprint arXiv:2108.09373.

[31] Xu, Y., Lee, H., Chen, D., Hechtman, B., Huang, Y., Joshi, R., Krikun, M., Lepikhin, D., Ly, A., Maggioni, M., Pang, R., Shazeer, N., Wang, S., Wang, T., Wu, Y., & Chen, Z. (2021). GSPMD: General and Scalable Parallelization for ML Computation Graphs. arXiv preprint arXiv:2105.04663.

[32] Smith, S., Patwary, M., et al. (2022). Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B: A Large-Scale Generative Language Model. arXiv preprint arXiv:2201.11990.

[33] Rajbhandari, S., et al. (2022). DeepSpeed-MoE: Advancing Mixture-of-Experts Inference. arXiv preprint arXiv:2201.05596.

[34] (2025). AXLearn: Modular Large Model Training on Heterogeneous Infrastructure. arXiv preprint arXiv:2507.05411v2.

[35] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., *et al.* (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*.

[36] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M.-W. (2020). REALM: Retrieval-Augmented Language Model Pre-Training. In *ICML* (PMLR 119).

[37] Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., *et al.* (2022). Improving Language Models by Retrieving from Trillions of Tokens (RETRO). In *ICML* (PMLR 162).

[38] Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., & Lewis, M. (2020). Generalization through Memorization: Nearest Neighbor Language Models. In *ICLR*.

[39] Wu, Y., Rabe, M., Hutchins, D., & Szegedy, C. (2022). Memorizing Transformers. In *ICLR*.

[40] Asai, A., Wang, X., Min, S., & Hajishirzi, H. (2023). Self-RAG: Learning to Retrieve, Generate, and Critique for Improved Language Modeling. *arXiv:2310.11511*.

[41] Islam, S. B., Parvez, M. R., *et al.* (2024). OPEN-RAG: Enhanced Retrieval-Augmented Reasoning with Open-Source LLMs. In *Findings of EMNLP 2024*.

[42] Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. In *CHI 2023*.

[43] Wang, G., Xie, Y., Jiang, Y., *et al.* (2023). Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv:2305.16291*.

[44] Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., & Yao, S. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning. In *NeurIPS 2023*.

[45] Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., & Gonzalez, J. E. (2023). MemGPT: Towards LLMs as Operating Systems. *arXiv:2310.08560*.

[46] Wang, W., Chen, H., Zhang, Y., *et al.* (2023). Augmenting Language Models with Long-Term Memory (LongMem). In *NeurIPS 2023*.

[47] Zhang, Z., Sun, H., & Zhang, Y. (2024). A Survey on the Memory Mechanism of Large Language Model based Agents. *arXiv:2404.13501*.

[48] Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., & Fang, Y. (2024). Evaluating Very Long-Term Conversational Memory of LLM Agents. In *ACL 2024*.

[49] Zhong, W., Zhang, X., & Wang, W. (2024). Enhancing Large Language Models with Long-Term Memory (MemoryBank). In *AAAI 2024*.

[50] Chen, H., *et al.* (2025). A-MEM: Agentic Memory for LLM Agents. *arXiv:2502.12110*.

[51] Wu, Q., Gao, S., *et al.* (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. *arXiv:2308.08155*.

[52] Li, G., Hammoud, H. A. A. K., Itani, H., Khizbullin, D., & Ghanem, B. (2023). CAMEL: Communicative Agents for "Mind" Exploration of LLM Society. *arXiv:2303.17760*.

[53] Breck, E., Polyzotis, N., Roy, S., Whang, S. E., & Zinkevich, M. (2019). Data Validation for Machine Learning. In *MLSys 2019*.

[54] Baylor, D., Breck, E., *et al.* (2017). TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. In *KDD 2017*.

[55] Sculley, D., Holt, G., Golovin, D., *et al.* (2015). Hidden Technical Debt in Machine Learning Systems. In *NeurIPS 2015*.

[56] Lewis, G. A., Echevería, S., & Lewis, B. A. (2022). A Step Towards Realistic Drift Detection in Production ML Systems. SEI/CMU Technical Report.

[57] Vajda, D. L., *et al.* (2024). Machine learning-based real-time anomaly detection using periodicity and prediction errors. *Scientific Reports* 14, 18702.

[58] Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A Distributed Messaging System for Log Processing. In *NetDB 2011*.

[59] Carbone, P., Ewen, S., Richter, S., *et al.* (2017). State Management in Apache Flink: Consistent Stateful Distributed Stream Processing. *PVLDB*, 10(12), 1718–1729.