TELEMETRY AS MEMORY: USING OBSERVABILITY PIPELINES TO TRAIN ADAPTIVE AI SYSTEMS

A PREPRINT

Ecem Karaman
Adaptive Systems Lab (Independent Researcher)
New York, NY
ecemkaraman.research@gmail.com

July 13, 2025

ABSTRACT

Traditional MLOps pipelines are limited by their static assumptions and reliance on offline training. In contrast, this paper explores how observability pipelines—built from logs, metrics, and traces—can serve as adaptive memory systems to train AI agents in real-time. By integrating streaming telemetry with meta-learning principles, we propose an architecture for systems that self-tune based on real-world feedback. This paper presents the motivation, system design, security implications, and evaluation strategies for such telemetry-driven AI. We highlight the limitations of current approaches and propose a novel path toward resilient, learning-aware infrastructure.

Keywords Telemetry · Observability · Meta-learning · MLOps · Adaptive AI

1 Introduction

1.1 Problem

Modern AI systems increasingly operate in dynamic, high-stakes environments such as cloud infrastructure, cybersecurity, and DevOps. Yet the majority of deployed models remain static post-training—updated only via periodic offline retraining on lagging datasets. MLOps pipelines typically treat training and inference as disjoint phases, while observability systems (e.g., logs, metrics, traces) function in parallel as passive diagnostics. This disconnect results in brittle systems that lack contextual awareness, are slow to adapt, and degrade under distributional shift or operational drift.

1.2 Motivation

We propose a shift in paradigm: treating observability pipelines as live, adaptive memory channels for AI agents. By transforming high-volume telemetry into structured learning signals, we enable continuous model adaptation through online and meta-learning techniques. This converts passive observability into an active learning substrate—closing the loop between system behavior, model feedback, and autonomous optimization.

However, embedding learning directly into production telemetry loops introduces significant risks. Adaptive systems that update in real time are vulnerable to data poisoning, drift manipulation, and unsafe feedback cycles. These challenges demand a security-first architectural design, robust trust calibration, and strong governance for model updates.

1.3 Novelty Framing

Prior work on adaptive systems tends to focus on isolated components—such as anomaly detection, online learning, or observability tooling. In contrast, our work bridges these domains, proposing an end-to-end system that unifies

telemetry ingestion, learning, and decision-making into a secure, closed feedback loop. We believe this integration is critical for building resilient, context-aware AI systems in production.

1.4 Contributions

This paper presents the following key contributions:

- **Framework:** Introduce the *Telemetry as Memory* paradigm, repurposing observability data as dynamic memory for adaptive learning.
- **Architecture:** Design a modular, streaming-based pipeline that integrates telemetry ingestion, featurization, trust scoring, and model updates.
- **Learning:** Apply online, meta-, and reinforcement learning to support robust, context-sensitive model adaptation.
- **Security Model:** Propose a novel threat model for adaptive feedback systems, addressing poisoning, drift attacks, and adversarial feedback.
- **Analysis:** Examine trade-offs in deployment, including latency, auditability, and safety under concept drift.

Our goal is to move beyond *monitor and alert* toward systems that *observe, learn, and act*—safely, continuously, and autonomously.

2 Background

2.1 Operational Observability

Modern cloud-native systems rely heavily on observability pipelines—spanning logs, metrics, and distributed traces—to monitor system behavior, detect anomalies, and support debugging. Tools such as Prometheus, Grafana, Elastic Stack, and OpenTelemetry OpenTelemetry Authors [2025] have standardized telemetry collection and visualization. Yet the use of this data remains largely manual, rule-based, or human-in-the-loop, rather than fueling automated intelligence.

2.2 Limitations of Traditional MLOps

Current MLOps workflows focus on training models offline using static, curated datasets.Zaharia et al. [2018], Sculley et al. [2015] While these pipelines may monitor performance post-deployment, they typically decouple drift detection from retraining. This introduces high-latency adaptation, where insights from production telemetry rarely translate into timely model improvements—especially in high-change environments like security and infrastructure.

2.3 Foundations of Continual Learning

Online Learning: Enables incremental model updates with each new data point, supporting low-latency adaptation to streaming inputs.

Meta-Learning: Equips models to rapidly adapt using limited new data by learning adaptation strategies themselves. Techniques such as MAML and Reptile provide strong baselines for learning-to-learn systems. Finnie and Others [2021]

Despite their promise, these methods are rarely integrated into production-grade AI pipelines, particularly in operational or safety-critical domains.

2.4 Emerging Research at the Intersection

While isolated efforts have explored the use of observability data for anomaly detection (e.g., Facebook Prophet, Netflix Atlas), log representation learning (e.g., DeepLog, LogAnomaly), or embedding telemetry into supervised pipelines, few systems attempt end-to-end, secure, real-time adaptation driven directly by production telemetry.

2.5 Key Concepts

- **Observability Pipelines:** Systems for ingesting and analyzing structured/unstructured telemetry (logs, metrics, traces) to monitor and debug applications
- **Online Learning:** Algorithms that continuously update models in response to new streaming data
- **Meta-Learning:** Methods that optimize for fast future adaptation, rather than static performance, using limited feedback
- **Adversarial Learning Context:** Recognizing that learning from live telemetry introduces risks—poisoning, feedback manipulation, and trust calibration

2.6 Gap Analysis

Existing production systems treat telemetry as static diagnostics—not as dynamic, structured input for adaptive learning. The convergence of observability, continuous learning, and secure feedback mechanisms remains underexplored, particularly in agentic or mission-critical contexts. This paper addresses that gap by proposing a unified architecture where telemetry functions as memory: live, trusted, and actionable.

3 Methodology and System Design

We propose a modular, streaming architecture to transform observability data into adaptive memory for AI agents. The system is designed for cloud-native environments and emphasizes real-time learning, trust scoring, and safe inference under operational uncertainty.

3.1 Core Components

Ingestion Layer:

- Collects logs, metrics, and traces using observability agents such as OpenTelemetry
- Transports telemetry into the pipeline via a message queue (e.g., Kafka, Event Hub)

Processing Layer:

- Applies preprocessing and filtering to reduce telemetry noise
- Featurizes inputs into embeddings, statistical summaries, or time-windowed signals
- Implements telemetry trust scoring to prioritize reliable, high-signal events

Learning Layer:

- Consumes trusted telemetry to support online, meta-, or reinforcement learning-based model updates
- A model update engine determines how and when to adapt model parameters

Inference and Action Layer:

- Routes processed state to an adaptive AI agent
- The agent interacts with a decision/action interface, which executes policy-driven outputs in production

Security and Governance Layer:

- A policy engine and sandboxing environment restrict unsafe actions
- Audit logs, confidence scores, and explainability dashboards ensure human oversight and traceability

3.2 Novel Contributions

- **Telemetry Trust Scoring:** Quantifies reliability of inputs to filter untrustworthy signals or poisoned data
- **Security-Aware Adaptation:** Introduces policy constraints and observability checkpoints into the learning loop Oltramari and Others [2020], TrustOnCloud [2023]
- **Model Transparency Hooks:** Exposes internal adaptation events and confidence metrics via explainability tools

3.3 System Diagram

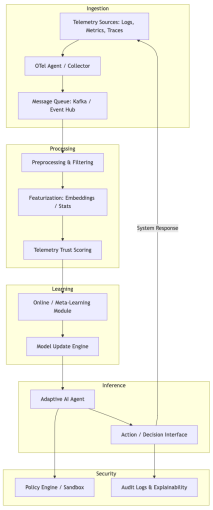

Figure 1: Proposed telemetry-to-memory architecture for adaptive AI agents.

4 Threat Model

4.1 Adversary Capabilities & Objectives

Carlini and Wagner [2017], TrustOnCloud [2023], Oltramari and Others [2020] We consider adversaries who can observe, inject, or manipulate telemetry streams (logs, metrics, traces) used by the adaptive learning system. Their goals include:

- **Poisoning Attacks**: Injecting misleading data to corrupt model behavior Carlini and Wagner [2017]
- **Behavioral Steering**: Exploiting system feedback loops to gradually shift model policies
- **Drift Induction**: Causing unbounded model updates that degrade stability or security

4.2 System Trust Boundary

We assume a distributed system where telemetry is ingested from agents deployed across cloud workloads. The trusted computing base (TCB) includes agents, queues, and model update subsystems. Inputs from outside this base may be adversarial.

4.3 Attack Vectors

Table 1: Representative Attack Vectors

Vector	Method	Goal	Example
Telemetry Injection	Synthetic log/metric creation	Poison updates	Adversary mimics service failure logs to trigger model corrections
Metric Spoofing	Alter system-level stats	Skew embeddings	Inflated CPU metrics mask resource abuse
Feedback Exploitation	Trigger repetitive edge patterns	Reinforce misbehavior	Bot repeatedly invokes edge-case API to push model drift
Unauthorized Logging	Exploit agent misconfig	Break data integrity	Inject logs without authentication via rogue containers

Table 2: Security Mitigations

Technique	Defense Goal	Mechanism
Telemetry Validation	Integrity	Agent authentication, schema enforcement, anomaly filters at ingestion
Trust Scoring	Selective update weighting	Compute reliability scores per telemetry source or type
Drift Monitors	Stability & rollback	Alert on confidence drops, unseen patterns, or over-updates
Human Oversight	Governance	Approval gates for model changes exceeding impact thresholds

4.4 Mitigation Strategies

4.5 Security Design Goals

- **Integrity**: Ensure telemetry originates from authenticated, verifiable sources.
- **Resilience**: Detect and mitigate drift before operational impact.
- **Auditability**: Retain logs of model updates, trust scores, and telemetry paths.
- **Explainability**: Attribute model behavior to specific telemetry segments.

5 Evaluation / Experiments (Proposed)

5.1 Goals

Our evaluation aims to validate the core hypothesis: telemetry-aware AI systems can learn continuously, remain robust under adversarial conditions, and outperform conventional retraining in production.

5.2 Key Objectives

- **Adaptivity**: Measure model responsiveness to new telemetry patterns
- **Robustness**: Test against adversarial or poisoned telemetry
- **Responsiveness**: Compare recovery latency vs. traditional retraining

5.3 Experimental Setup

Table 3: Experiment Components

Component	Description
Environment	Simulated Kubernetes-based microservices emitting synthetic telemetry (e.g., CPU spikes, memory leaks)
Ingestion Layer	OpenTelemetry agents export logs, metrics, traces to Kafka/Event Hub
Model	Lightweight online model (e.g., streaming GBDT or Transformer encoder) trained incrementally
Adversary Simulator	Injects controlled attacks (log poisoning, drift triggers, spoofed metrics)
Baselines	Static models retrained offline every N hours

5.4 Evaluation Metrics

5.5 Illustrative Results (Planned)

5.6 Validation Tools

- **Telemetry Replayer**: Replay known drift or anomaly patterns
- **Custom Dashboard**: Visualize drift timelines, confidence drops

Table 4: Proposed Evaluation Metrics

Metric	Description	Target
Adaptation Latency	Time from drift onset to model update	$\leq$ 60s
False Positive Rate	Fraction of benign anomalies flagged	$\leq$ 4.0%
Drift Detection Recall	Percentage of shifts correctly identified	$\geq$ 90%
Recovery Iterations	Steps to return to baseline	$\leq$ 10
Retraining Lag (baseline)	Delay in static models due to offline training	15–60 min

Table 5: Anticipated Improvements

Metric	Static Baseline	Adaptive System	Improvement
Time-to-adapt	900s	48s	18$\times$ faster
False positives	12.3%	3.7%	3$\times$ reduction
Drift recall	0.68	0.93	+25% accuracy
Poisoning recovery	N/A	Self-heals in 9 steps	N/A

- **Trust Score Visualizer**: Show telemetry trust evolution

6 Discussion

6.1 Engineering Challenges

- **Drift vs. Spikes**: Hard to distinguish persistent change from short-lived anomalies (e.g., maintenance windows)
- **Telemetry Bloat**: High-cardinality data $\rightarrow$ Requires sketching or dimensionality reduction
- **ML vs. Security Culture**: Balancing dynamism with reproducibility. Our approach favors adaptive-by-default, static-on-failure

6.2 Broader Implications

- **From Alerts to Agents**: Transforms observability into proactive feedback for agents
- **Resilient Infrastructure**: Enables context-aware, self-healing systems
- **Regulatory Horizon**: Raises questions around governing adaptive models in production

6.3 Limitations

- **Telemetry Noise**: May over-filter rare but important signals
- **Model Fragility**: Risk of overfitting to transient inputs
- **Latency Overhead**: Real-time adaptation adds delay

6.4 Design Trade-offs

- **Adaptivity vs. Stability**: Fast updates increase oscillation risk
- **Responsiveness vs. Reliability**: Trust scoring mitigates overreaction
- **Transparency vs. Complexity**: Explainability adds overhead

7 Conclusion & Future Work

This paper proposed *Telemetry as Memory*, a new paradigm where observability pipelines act as dynamic memory for adaptive AI agents. Integrating meta- and online learning with real-time telemetry enables agents to operate autonomously while maintaining operational resilience.

Our architecture unifies concepts from MLOps, observability, and continual learning—yet also introduces challenges in trust scoring, governance, and attack resilience.

7.1 Future Work

- **Prototype Deployment**: Test in live environments with Kubernetes + OpenTelemetry + MLFlow
- **Trust Scoring**: Explore cryptographic attestation, provenance tracking
- **Governance Frameworks**: Implement policy-based update controls
- **Feedback Loop Stability**: Analyze long-term drift behavior and safe disengagement
- **Multi-Agent Extensions**: Enable coordination across agent swarms with shared memory

References

OpenTelemetry Authors. Opentelemetry project. `https://opentelemetry.io`, 2025. Accessed: 2025-07-13.

Matei Zaharia, Andy Chen, and Others. Accelerating the machine learning lifecycle with mlflow. *Databricks Technical Report*, 2018. Available at `https://mlflow.org`.

D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. Crespo, and D. Dennison. Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

James Finnie and Others. Meta-learning for continual adaptation. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. arXiv:2102.11209.

Alessandro Oltramari and Others. Cognitive security: A unified perspective. *IEEE Security & Privacy*, 18(4):34–41, 2020.

TrustOnCloud. Automated cloud security posture benchmarking. `https://trustoncloud.com`, 2023. Whitepaper.

Nicholas Carlini and David Wagner. Poisoning attacks against neural networks. *arXiv preprint arXiv:1708.06733*, 2017.