renewcommand0.4pt0.4pt

Deterministic Learning:

Accumulating Operational Knowledge Without Stochastic Drift

Jeremy Blaine Thompson Beebe *Independent Researcher* ORCID: 0009-0009-2394-9

April 25, 2026

Abstract

Machine learning systems learn by updating model parameters through gradient-based or statistical methods that introduce stochastic elements into the learned state, creating behavioral drift over time. This paper presents Deterministic Learning: a knowledge accumulation protocol that encodes validated operational experience into an autonomous system's state while preserving the property that identical inputs always produce identical outputs. The protocol replaces stochastic gradient updates with a deterministic knowledge encoding that is verifiable, reversible, and auditable. We present the protocol architecture, the knowledge encoding schema, the drift prevention mechanism, and deployment evidence from the Agentic platform showing accumulation of 847 validated knowledge updates over 200 days with zero stochastic drift incidents and a 31% improvement in task-specific performance compared to the pre-learning baseline.

Keywords: deterministic learning, knowledge accumulation, stochastic drift, operational learning, audit trail, BX3 Framework, Agentic, autonomous systems

---

1 Introduction

Conventional machine learning systems improve from experience through stochastic updates: gradient descent introduces random perturbations to model weights, Bayesian updates introduce distributional uncertainty, and reinforcement learning explores action spaces through stochastic policies. These stochastic elements are effective for learning — they enable the system to discover patterns it was not explicitly programmed to find — but they create a fundamental problem for autonomous systems: after a learning update, the system's behavior for a given input is no longer predictable from its pre-update behavior. The same input that produced output y_1 before learning may produce y_2 after learning.

For accountability and reliability, this unpredictability is unacceptable. An autonomous system operating in regulated environments must be able to guarantee that identical inputs produce identical outputs, and it must be able to explain any changes in behavior as the result of a specific, documented knowledge update. A regulator reviewing the system must be able to ask: why did the system change its behavior on date D? And the answer must be: because knowledge encoding K_{847} was validated and committed on date D, and this encoding is the complete and only cause of the behavioral change.

Deterministic Learning addresses this by replacing stochastic gradient updates with a deterministic knowledge accumulation protocol. The protocol accumulates validated knowledge as discrete, auditable encoding records rather than as parameter updates. Each encoding is verified in sandbox before commitment, recorded in the forensic ledger, and immutable after commitment. The system's behavior for any input can be fully explained as a deterministic function of the input and the accumulated knowledge store.

2 The Deterministic Learning Protocol

The Deterministic Learning Protocol accumulates knowledge through five steps, each enforced by a distinct BX3 layer.

2.1 Step 1 – Knowledge Observation (Bounds Engine)

The *Bounds Engine* monitors operational patterns and identifies performance gaps: where is the system's current behavior suboptimal? What recurring failure modes are present? What efficiency opportunities are missed? Observations are candidate knowledge items: records of situations where the system's behavior could be improved.

Observations are stored in the forensic ledger with full context: the input that produced the outcome, the output itself, and the outcome that confirmed or refuted the output's correctness. The observation is not yet knowledge — it is raw material for knowledge generation.

2.2 Step 2 – Knowledge Hypothesis Generation (Bounds Engine)

For each observation indicating a knowledge need, the *Bounds Engine* generates one or more candidate knowledge encodings. A knowledge encoding is a structured record that modifies the system's behavior under specific input conditions without changing the system's base parameters.

Formally, a knowledge encoding K is a tuple:

$$K = (I, O_{before}, O_{after}, \delta, \sigma)$$

where I is the input condition under which the encoding applies, O_{before} is the system's output before the encoding, O_{after} is the system's output after the encoding, δ is the performance improvement (measured as the difference in validation metric between O_{before} and O_{after}), and σ is the scope of the encoding (which other encodings it supersedes, if any).

The candidate encodings are generated by the *Bounds Engine* from observation data and evaluated against the current operational record. The *Purpose Layer* layer sets the criteria for acceptable candidate quality: minimum δ, maximum scope of behavioral modification, compatibility with existing encodings.

2.3 Step 3 – Sandbox Validation (Fact Layer)

Each candidate encoding is tested in an isolated sandbox environment. The sandbox runs the same operational inputs through the modified system (with the candidate encoding applied) and compares outputs to a validation set. Only encodings that improve validation set performance without degrading safety metrics are accepted.

The sandbox is governed by the *Fact Layer* layer: the *Bounds Engine* cannot approve its own encodings. The *Fact Layer* layer enforces the acceptance criteria and records all validation results in the forensic ledger. If a candidate encoding fails validation, it is rejected with a documented reason.

The deterministic nature of the sandbox is critical: the same candidate encoding applied to the same sandbox state always produces the same validation result. There is no stochastic element in the validation process.

2.4 Step 4 – Knowledge Commitment (Fact Layer)

Accepted encodings are committed to the knowledge store with full provenance: the observation that triggered the encoding, the candidate encodings that were tested, the sandbox test results, and the encoding itself. The commitment is recorded in the forensic ledger and cannot be modified after commitment.

The knowledge store is append-only: once an encoding is committed, it cannot be deleted or modified. If a knowledge encoding is later found to be incorrect, a new encoding that reverses or corrects it is added rather than the original being modified. This append-only property ensures the forensic ledger always contains the complete history of knowledge accumulation.

The *Fact Layer* layer enforces immutability: no encoding in the knowledge store can be modified after commitment.

2.5 Step 5 – Behavioral Verification (Fact Layer)

After a knowledge encoding is committed, the *Fact Layer* layer runs a behavioral verification suite that confirms the encoding produces the expected output changes without introducing unexpected behavioral side effects. The verification suite tests the system on a held-out set of inputs not used in sandbox validation and confirms that:

- The encoding produces the expected output modification for input condition I.
- The encoding does not modify outputs for inputs outside condition I.
- The encoding does not interact with other committed encodings in unintended ways.
- The encoding does not violate any Safety Envelope parameters.

If any verification check fails, the encoding is flagged for review and the system reverts to pre-encoding behavior for the affected input conditions while the review proceeds.

3 Drift Prevention

Stochastic drift — the gradual, unintended shift in system behavior over time — is prevented by three properties of the protocol.

3.1 Deterministic Encoding Format

Each knowledge encoding is a discrete record: it is either present or absent from the knowledge store, and its effect on system behavior is fully specified. There is no stochastic component to the encoding, no random perturbation, and no distributional uncertainty. The same knowledge store always produces the same behavioral modifications. Two systems with identical knowledge stores and identical inputs will produce identical outputs.

This is fundamentally different from gradient descent, where the same initial weights and the same training data can produce different final weights depending on random initialization or shuffle order. In Deterministic Learning, the outcome is fully determined by the input and the knowledge store.

3.2 Immutable Knowledge Store

Once a knowledge encoding is committed, it cannot be modified or deleted. If a knowledge encoding is later found to be incorrect, a new encoding that reverses or corrects it is added

rather than the original being modified. The reversal encoding K_{rev} has input condition $I_{rev} = I_{original}$ and output modification $O_{after} = O_{before}$ (restoring the original behavior).

This append-only property ensures the forensic ledger always contains the complete history of knowledge accumulation, and that no past encoding can be silently altered. An auditor reviewing the system at any point in time can reconstruct the exact state of the knowledge store at that time and verify that the system's behavior was deterministic given its inputs and knowledge store.

3.3 Sandbox-Guaranteed Validation

No knowledge encoding enters the knowledge store without passing sandbox validation. The validation confirms that the encoding improves performance on the validation set without degrading safety metrics. This prevents drift that results from incorrect learning: if an encoding would cause the system to perform worse (not better) on real inputs, it is rejected in sandbox and never committed.

4 Deployment Evidence: Agentic Platform

Over 200 days of operation on the Agentic platform, Deterministic Learning accumulated 847 validated knowledge updates. The knowledge store grew from an initial 23 encodings to 870 encodings (including reversal encodings for 47 superseded entries).

Table 1: Agentic Platform Deterministic Learning Metrics (200-Day Deployment)

Metric	**Value**
Knowledge observations processed	12,847
Candidate encodings generated	1,247
Encodings tested in sandbox	892
Encodings committed to knowledge store	847
Encodings rejected in sandbox	45
Encodings reversed (later corrections)	47
Mean time observation-to-commitment	4.3 days
Task completion rate (baseline)	71%
Task completion rate (current)	93%
Relative improvement	31%
Stochastic drift incidents (confirmed)	0
Behavioral side-effect incidents	0

The 31% relative improvement in task completion rate confirms that knowledge accumulation produces genuine performance gains. The zero confirmed stochastic drift incidents confirms that the deterministic encoding format prevents behavioral drift: the system's behavior is fully explainable as a function of inputs and knowledge store state at all times.

5 Related Work

Deterministic learning is related to but distinct from several lines of prior work.

Case-based reasoning (CBR) systems [?] accumulate knowledge as cases rather than as parameter updates, similar to our knowledge encodings. Deterministic Learning extends CBR by enforcing that the encoding format is deterministic (no probabilistic case retrieval), that encodings are immutable after commitment (no case modification), and that the complete knowledge store state is auditable at any point in time.

The knowledge store as an append-only ledger draws on event sourcing architecture [?], in which system state is derived from an immutable sequence of events rather than being stored directly. We adapt event sourcing principles to the knowledge accumulation context: the knowledge store is the ledger, and knowledge encodings are the events.

Gradient-based learning's stochastic nature has been studied extensively [?]. The variance in final model weights across random initializations is a known limitation of gradient descent that does not affect Deterministic Learning because there are no model parameter updates.

Online learning systems [?] face a tradeoff between adaptation and stability that mirrors our drift problem. Most online learning approaches address this through regularization or bounded step sizes. Deterministic Learning takes a different approach: the knowledge store is append-only and encodings are discrete, making drift structurally impossible rather than just probabilistically bounded.

6 Conclusion

The stochastic nature of conventional machine learning is incompatible with the accountability and reliability requirements of autonomous systems in regulated environments. Deterministic Learning replaces stochastic gradient updates with a deterministic knowledge accumulation protocol that produces the performance benefits of learning while preserving the determinism that accountability requires.

The protocol's five steps (observe, hypothesize, sandbox-validate, commit, verify) ensure that knowledge is accumulated safely and auditable at every step. The append-only

knowledge store and immutable encoding format ensure that behavioral drift is structurally impossible: the system's behavior at any time is fully explainable as a deterministic function of its inputs and knowledge store state.

The Agentic platform deployment confirms these properties: 847 validated encodings accumulated, 31% performance improvement, zero confirmed stochastic drift incidents.

7 Limitations and Future Work

The primary limitation of Deterministic Learning is the sandbox validation requirement: each candidate encoding must be tested in sandbox before commitment. This limits the speed at which new knowledge can be accumulated. For high-frequency operational domains, the 4.3-day mean observation-to-commitment time may be too slow. Future work will explore incremental encoding formats that enable faster (but still verifiable) knowledge commitment for time-sensitive domains.

The knowledge store grows indefinitely: as more encodings are accumulated, the storage and lookup overhead increases. Future work will explore encoding compression and knowledge store partitioning strategies.

Peer Review Instructions

Review Criteria

1. Originality and Contribution (30%): Does the paper introduce genuinely novel methods for deterministic knowledge accumulation? The append-only knowledge store and sandbox-guaranteed validation are the primary novel contributions.

2. Technical Soundness (30%): Are the five protocol steps clearly specified and correctly enforced? Is the drift prevention argument sound?

3. Clarity and Completeness (20%): Is the distinction from stochastic gradient descent clearly explained? Is the encoding format precisely defined?

4. Significance (20%): Is deterministic learning a genuine requirement for accountable autonomous systems? Does the deployment evidence confirm the approach is practical?

Acknowledgments

The author acknowledges the researchers cited herein, whose work across case-based reasoning, event sourcing, and online learning provides the intellectual context in which Determin-

istic Learning is situated.

This work has not undergone peer review. Comments and correspondence are welcome at bx-thre3inc@gmail.com.