Harness-Layer Context Engineering as External Nested Learning for Deployed LLMs

Mike Morris

Abstract. Large language models are static after deployment — they cannot form new long-term memories or adapt to new domains without retraining. Behrouz et al. (2025) formalize this as "anterograde amnesia" and show that neural networks are hierarchies of associative memories operating at different timescales, with continual learning requiring multi-frequency memory consolidation inspired by neural oscillations. We demonstrate that these nested learning properties can be restored *externally* through a harness-layer architecture that provides multi-timescale memory, surprise-driven refinement, and confidence-weighted knowledge consolidation — without modifying model weights. We present five production systems implementing this architecture across compliance assessment, code assistance, and multi-agent coordination domains. Our strongest empirical result: domain-specific fine-tuning plateaus at 84% accuracy on a CMMC compliance evaluation benchmark, while the addition of external context injection at inference time achieves 100% on the same benchmark with zero additional training. We formalize the mapping between Nested Learning's frequency-band framework and harness-layer components, and argue that external nested learning is preferable to internal approaches for production systems operating in regulated environments.

---

1. Introduction

For decades, the central challenge of machine learning has been designing systems that learn from data and improve with experience (Pitts, 1943; McCulloch & Pitts, 1943). Deep learning has made extraordinary progress on this front, with large language models (LLMs) demonstrating remarkable capabilities across diverse tasks (Brown et al., 2020; Achiam et al., 2023; Comanici et al., 2025). Yet a fundamental problem remains: LLMs are largely static after deployment. They cannot continually acquire new knowledge, adapt to new domains, or learn from their mistakes without expensive retraining cycles that risk catastrophic forgetting (Akyürek et al., 2024; Eyuboglu et al., 2025).

Behrouz et al. (2025) recently formalized this limitation using the analogy of *anterograde amnesia* — a neurological condition where a person cannot form new long-term memories after the onset of the disorder. The patient experiences only the immediate present, with knowledge limited to a short window of current experience and whatever was stored before the condition began. Current LLMs suffer from a strikingly similar pattern: their knowledge is limited to the immediate context window and whatever was compressed into model parameters during pre-training, before the onset of "end of pre-training."

Their Nested Learning (NL) framework reveals that neural networks are not monolithic function approximators but hierarchies of *associative memories*, each compressing its own "context flow" at a different timescale. Optimizers like Adam and SGD with momentum are themselves associative memories that compress gradient information. Attention mechanisms are two-level optimization processes. The entire training procedure — from pre-training through in-context learning — can be decomposed into nested optimization problems operating at frequencies ranging from per-token (analogous to gamma brain waves at 30–150 Hz) through per-epoch (analogous to delta waves at 0.5–4 Hz).

This paper makes a simple but consequential observation: **if continual learning requires multi-timescale memory consolidation, and deployment freezes all internal timescales, then the missing timescales can be provided externally.** The brain itself suggests this architecture. Memory consolidation in the hippocampus is anatomically separate from reasoning in the cortex (Scoville & Milner, 1957; Foster et al., 2006). The hippocampus receives experiences, consolidates them across timescales, and injects consolidated knowledge back into cortical processing. The cortex reasons; the hippocampus remembers.

We build the same separation for LLMs. The *harness layer* — the software infrastructure surrounding a deployed model — serves as an external hippocampus, providing multi-timescale memory, surprise-driven refinement, and knowledge consolidation. The model receives augmented context through hook-based injection but does not know it has been augmented. This separation provides four properties that internal continual learning approaches cannot easily achieve: model-agnosticism (the same harness works across Claude, GPT, Gemma, and local models), auditability (the harness operates at an inspectable trust boundary), compliance safety (no CUI or sensitive data touches model weights), and zero-cost adaptation (new knowledge is available in seconds, not hours).

1.1 Contributions

1. A formal mapping between Nested Learning's multi-timescale frequency framework and harness-layer architecture components, establishing external nested learning as a theoretically grounded approach to post-deployment continual improvement.

2. Five production systems demonstrating external nested learning across compliance assessment, code assistance, multi-agent coordination, enterprise AI governance, and expert system construction — providing evidence that the architecture generalizes across domains.

3. An empirical comparison showing that external context injection at inference time achieves 100% accuracy on a domain-specific evaluation benchmark where fine-tuning alone plateaus at 84%, with a Nested Learning interpretation of why.

4. A compliance-safe architecture for continual learning in regulated

environments (CMMC, FedRAMP, HIPAA), where the separation between reasoning (model) and memory (harness) maps directly to data-boundary requirements.

---

2. Background

2.1 Nested Learning

Behrouz et al. (2025) present Nested Learning as a paradigm that decomposes machine learning models into sets of nested, multi-level optimization problems. Their key insights, which we build upon:

Associative memory formulation. Given keys $K \subseteq \mathbb{R}^{d_k}$ and values $V \subseteq \mathbb{R}^{d_v}$, an associative memory is an operator $M(\cdot)$ that maps keys to values. Training this operator via an objective $\hat{L}$ is the fundamental learning operation. All neural network components — including optimizers, attention layers, and feedforward networks — can be viewed as associative memories compressing their respective context flows.

Local Surprise Signal (LSS). Training a linear layer with backpropagation can be reformulated as building an associative memory that maps each data point to the error of its corresponding prediction — the "surprise" in representation space. This formulation translates the training phase into a process of acquiring memory that captures how surprising each observation is relative to the model's current knowledge.

Multi-timescale updates. Inspired by neural oscillations (brain waves), NL orders model components by their update frequency. High-frequency components (analogous to gamma waves, 30–150 Hz) handle fast adaptation — sensory processing, in-context learning at the token level. Low-frequency components (analogous to delta waves, 0.5–4 Hz) handle slow consolidation — pre-training updates that modify foundational representations. Components at different frequencies form a hierarchy where each level compresses its context flow and transfers knowledge to adjacent levels.

Continuum Memory System (CMS). NL generalizes the traditional binary long-term/short-term memory view into a *continuum* of memory frequencies. Higher-frequency neurons handle fast adaptation but forget quickly; lower-frequency neurons store persistent knowledge but update slowly. This distributed design means knowledge can be partially recovered even after apparent "forgetting," because traces persist across the frequency spectrum.

2.2 The Deployment Gap

All of these internal learning dynamics freeze at deployment. The only surviving adaptation mechanism is *in-context learning* (ICL) — the model's ability to adapt to new information presented within the context window (Brown et al., 2020). But ICL is ephemeral: it resets with every new conversation. A lesson learned in session N is forgotten by session N+1. The model cannot promote in-context

observations to any persistent memory layer.

Current approaches to bridging this gap have significant limitations:

- **Retrieval-Augmented Generation (RAG)** provides external knowledge at query time but operates at a single timescale — there is no consolidation, no surprise-driven refinement, no frequency hierarchy. RAG is a memory prosthetic, not a memory system.
- **Fine-tuning** modifies the model's lowest-frequency parameters (pre-training weights) but is expensive, risks catastrophic forgetting, and in regulated environments raises data-boundary concerns when sensitive information must touch model weights.
- **Prompt engineering** operates exclusively at the highest frequency (per-token context) with no persistence or consolidation.

None of these approaches restore the multi-timescale learning dynamics that NL identifies as essential for continual learning. We propose an architecture that does.

2.3 Context Engineering

Hua et al. (2025) formalize "context engineering" as a discipline with a four-era framework, progressing from context-as-translation (1990s HCI) through context-as-instruction (current LLM era) toward context-as-scenario (emerging agent era). Their survey identifies the need for a "semantic operating system capable of growing over time, much like a human mind… context is no longer passively accumulated, but is actively managed and evolved as a core element for cognition."

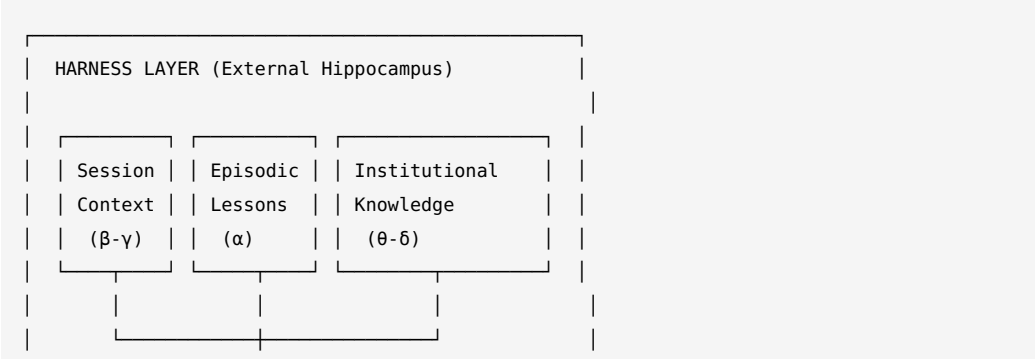
Our work provides a concrete architecture for this vision, grounded in the theoretical framework of Nested Learning.

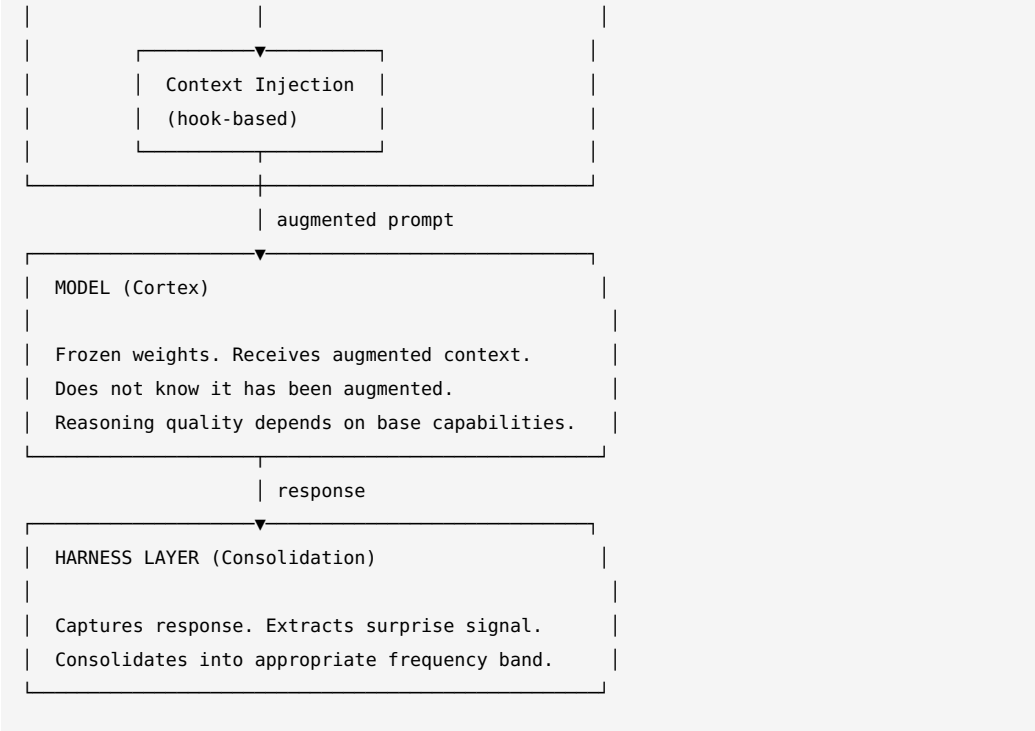
---

3. Architecture: External Nested Learning

3.1 The Hippocampus-Cortex Separation

The central architectural principle is separation of concerns between reasoning and memory, mirroring the brain's separation of cortical processing from hippocampal consolidation.





The model operates normally — it receives a prompt and produces a response. The harness intercepts at three points: before the prompt (to inject relevant context from memory), after the response (to capture the interaction for consolidation), and at session boundaries (to promote ephemeral context to persistent storage). These interception points are implemented as *hooks* — shell commands that execute in response to system events — rather than modifications to the model's inference pipeline.

3.2 Multi-Timescale Memory Hierarchy

We map NL's frequency bands directly to harness-layer storage components. Five bands emerge naturally from the architecture:

γ-band (30–150 Hz) — Sensory processing. The conversation itself: current prompt, tool outputs, streaming responses. Updates per-token. Session-scoped — evicted when the conversation ends. No compression; full fidelity.

β-band (12–30 Hz) — Active thinking. Working summaries and checkpoints that capture task state: what's being worked on, what's been tried, what failed. Updates per-task, persists days to weeks. Compressed from γ by extracting only what a future session would need to resume work.

α-band (8–12 Hz) — Relaxed awareness. Episodic lessons — discrete pieces of transferable knowledge. Example: "Traefik strips auth headers on /api routes when stripPrefix is enabled; fix: set preserveHostHeader=true." Updates per-session, persists weeks to months. Compressed from β by extracting only what generalizes beyond the specific session.

θ-band (4–8 Hz) — Memory consolidation. Mental models and institutional

knowledge: organized understanding of a domain. Example: “M365 GCC tenants require delegated auth; service principals are forbidden for CUI workloads.” Updates per-project, persists months to years. Compressed from α by clustering related lessons into coherent domain understanding.

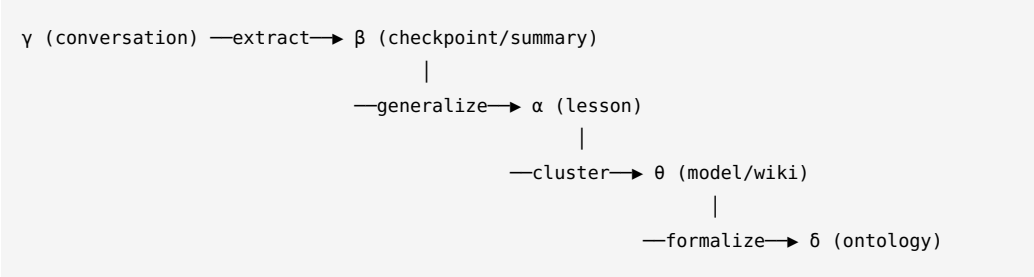
δ-band (0.5–4 Hz) — Deep consolidation. Domain ontologies and foundational schema: the structural knowledge that shapes all higher-frequency processing. Example: the OSCAL control framework (1,955 controls) or the VRDM type system. Updates per-version. Persistent. Changes only when the underlying framework changes.

Each band has a distinct storage format, compression strategy, and retrieval mechanism:

- **γ-band** storage is the raw conversation. No compression. Full fidelity. Evicted at session end.
- **β-band** storage is structured checkpoint documents: task description, current state, completed steps, next actions, key decisions. Compressed from γ by extracting only what a future session would need to resume work.
- **α-band** storage is discrete lessons: a title, content body, severity level, and tags. Each lesson captures one piece of transferable knowledge. Compressed from β by extracting only what generalizes beyond the specific session.
- **θ-band** storage is organized knowledge: wiki entries, mental models, experience logs. Compressed from α by clustering related lessons into coherent domain understanding.
- **δ-band** storage is structural: domain ontologies, type systems, formal schemas. These change rarely and represent foundational knowledge that shapes all higher-frequency processing.

3.3 Consolidation Pipeline

Knowledge flows downward through the hierarchy via a consolidation pipeline analogous to hippocampal memory consolidation:



Each transition is a compression step — NL’s “context flow” — where information is transformed from higher-fidelity, higher-frequency representations to lower-fidelity, lower-frequency, more durable representations. The consolidation is lossy by design: not every conversation becomes a checkpoint, not every checkpoint yields a lesson, and not every lesson modifies a mental model.

Critically, surprise signals flow *upward* through the hierarchy. When a model

response contradicts a θ-band mental model, this generates a correction at α-band that may eventually modify the θ-band knowledge. This bidirectional flow — consolidation downward, surprise upward — is the mechanism by which the system continually improves.

3.4 Surprise-Driven Refinement

Behrouz et al. show that training is fundamentally a process of memorizing the Local Surprise Signal — how surprising each observation is relative to the model's current predictions. We implement an external analog:

Definition (External Surprise Signal). An external surprise event occurs when the model's output contradicts established knowledge in the harness memory, when a human corrects the model's response, or when an automated verifier detects an error. Each surprise event is captured as a *correction* at α-band and tagged with the memory bands it affects.

The *Praxis refinement flywheel* (§4.2) operationalizes this: every incorrect response generates a correction that flows to the appropriate frequency band. Over time, the density of corrections in a domain indicates areas where the model's internal knowledge diverges most from ground truth — exactly the regions where external memory injection provides the most value.

3.5 Confidence-Weighted Knowledge

Not all knowledge is equally reliable. We assign confidence scores to knowledge assertions, drawing on the Knowledge Node Description Language (KNDL) framework for confidence-native assertions with temporal decay:

- Confidence ranges from 0.0 (unverified) to 1.0 (repeatedly confirmed).
- Confidence *decays* over time for uncorroborated knowledge — an assertion confirmed six months ago is less certain today than one confirmed yesterday.
- High-confidence knowledge updates slowly (low frequency); uncertain knowledge updates quickly (high frequency).

This maps directly to NL's multi-frequency update rates: **confidence is the frequency selector.** A high-confidence lesson behaves like a θ-band memory — persistent, slow to change, foundational. A low-confidence lesson behaves like a β-band memory — transient, rapidly updated, subject to revision. As confidence accumulates through repeated confirmation, knowledge naturally migrates from higher to lower frequency bands without explicit promotion logic.

3.6 Hook-Based Integration

The harness integrates with the model through system-level hooks rather than API modifications:

- **SessionStart hook:** Fires when a new conversation begins. Loads relevant checkpoints (β), searches lessons (α), and injects institutional context (θ) based on the session's declared task. This is the equivalent of the

hippocampus activating relevant memory traces at the start of a cognitive task.

- **UserPromptSubmit hook:** Fires before each user turn is sent to the model. Searches the memory hierarchy using the prompt as a query and injects relevant context. This provides per-turn memory augmentation at γ-band frequency.
- **Stop hook:** Fires at session end. Extracts key information from the conversation (γ) and consolidates it into checkpoints (β) and lessons (α). This is the equivalent of hippocampal replay during sleep — consolidating the day's experiences into more durable memory.

The model never sees the hook infrastructure. It receives a prompt that happens to contain relevant context, produces a response, and the harness captures the result. This "cognitive transparency" is a feature, not a limitation: it means the harness can be upgraded, replaced, or removed without any change to the model, and the model's behavior remains interpretable in terms of its inputs and outputs without hidden internal state modifications.

---

4. Production Systems

We present five systems that implement external nested learning. Each arose from practical engineering needs; the theoretical connection to NL was established post hoc when Behrouz et al. (2025) provided the formal framework.

4.1 Nellie: Hook-Based Context Engineering Middleware

Nellie is a persistent memory server, implemented in Rust, that provides multi-timescale storage for AI coding assistants. It integrates with Claude Code via the hook system described in §3.6.

Architecture. Nellie maintains four storage layers:

1. *Checkpoints* (β-band): Structured snapshots of working state — current task, completed steps, next actions, key decisions, relevant files. Agents save checkpoints every 10–15 minutes and at session boundaries.
2. *Lessons* (α-band): Discrete pieces of transferable knowledge with title, content, severity (critical/warning/info), and tags. Lessons are created when an agent encounters a surprising error, discovers a non-obvious pattern, or receives a correction from the user.
3. *Code index* (θ-band): Semantic embeddings of the codebase, updated incrementally on file changes. Provides structural understanding of the project.
4. *Knowledge graph* (δ-band): Relationships between concepts, files, lessons, and agents. Provides the ontological backbone for retrieval.

Measured outcomes. Context recovery time — the time from session start to productive work — is approximately 30 seconds with Nellie (loading checkpoints and relevant lessons) compared to approximately 10 minutes without (manually reconstructing context from files, git history, and memory). Token consumption per completed task is reduced by approximately 50%, as measured by comparing monthly API costs before and after adoption with equivalent workload intensity.

Cross-agent transfer. Lessons saved by one agent (e.g., a compliance specialist) are immediately searchable by another agent (e.g., a code assistant) through the shared Nellie server. This provides inter-agent knowledge transfer at α-band without any model retraining — an external analog of NL's cross-level knowledge transfer.

4.2 Praxis: Three-Layer Expert System Flywheel

Praxis is an architecture for building expert systems from dense documentation, with a three-layer structure that maps directly to NL frequency bands:

Layer 1 — Foundation (δ-band). Structured domain corpus ingested once and updated rarely. For CMMC compliance: 1,955 NIST controls in OSCAL JSON format, chunked one-concept-per-document for clean retrieval boundaries. For mortgage underwriting: guideline sections from GSE handbooks. The foundation is the lowest-frequency layer — it changes only when the underlying regulatory framework changes.

Layer 2 — Institutional Knowledge (θ-band). Organization-specific expertise layered on the foundation. For CMMC: 320 assessment objectives mapped to specific Microsoft Graph API endpoints, with human-authored guidance on what assessors actually look for, common gaps organizations miss, and automation percentages per objective. This layer represents approximately one year of assessment expertise distilled into searchable, injectable context. It updates quarterly as assessment patterns evolve.

Layer 3 — Refinements (α-β band). Per-user and per-engagement corrections that accumulate with every interaction. When the system produces an incorrect assessment, the correction is captured and stored. Each correction refines the system's responses for similar future queries. This layer updates with every session.

The flywheel mechanism. Corrections at Layer 3 accumulate. When a pattern of corrections reveals a systematic gap in Layer 2, the institutional knowledge is updated. This is the external surprise signal driving consolidation from high-frequency to low-frequency storage. Month one, the system operates at Layer 1 accuracy (~85% on domain-specific queries). By month six, accumulated corrections at Layers 2–3 approach expert-level performance on frequently-encountered scenarios. By month twelve, new operators benefit from day-one quality because institutional knowledge lives in the system, not in any individual's memory.

Deployed instances. CMMC-Buddy (compliance assessment), J-Mo (mortgage underwriting), and three proof-of-concept deployments in real estate and construction compliance.

4.3 SallyPort: Fine-Tuning Meets External Memory

SallyPort is a CMMC 2.0 Level 2 compliance assessment tool that provides our strongest empirical evidence for external nested learning. It uses a fine-tuned Gemma 3 12B model for automated assessment of 320 CMMC objectives against Microsoft 365 tenant configurations.

Fine-tuning trajectory. Four rounds of LoRA fine-tuning on progressively refined training data revealed a hard accuracy ceiling:

Round 1 (v1): Gemma 3 12B with QLoRA at learning rate 2e-4. Complete mode collapse — the model predicted DEFICIENT for every input, yielding 0% useful accuracy. The aggressive learning rate overwhelmed the pre-trained representations.

Round 2 (v1-fix): Learning rate reduced to 5e-5, batch size increased to 4. Accuracy reached 62% on the full evaluation set. However, 19 of 50 evaluation items were free-text knowledge questions scored by exact string match — an impossible bar for generative output. On verdict-only evaluation (VERIFIED/DEFICIENT/INSUFFICIENT_DATA classification), accuracy was 100%.

Round 3 (v2): Expanded training data with verdict-only evaluation methodology. Aggregate accuracy reached 84% (42/50) across mixed verdict and knowledge questions. Additional data beyond this point provided marginal improvement.

Round 4 (v3): Further expanded training data. No accuracy improvement — still 84%. In some configurations, additional data *degraded* performance on previously-correct items.

The plateau at 84% resisted additional fine-tuning data. Adding more examples did not improve accuracy and in some configurations degraded it — the “sweet spot” phenomenon where a specific data volume produces optimal results and exceeding it disrupts learned representations.

External memory intervention. We constructed a static context map: 640KB of pre-computed, per-objective decision guides containing verdict criteria, key evidence fields, numeric thresholds, and cross-domain trap warnings. At inference time, the relevant guide is injected into the prompt alongside the assessment data.

The result was stark. The fine-tuned model alone (v3) achieved 84% accuracy (42/50) after four training rounds totaling approximately 692 minutes of GPU time on an Apple M4 Pro, at roughly $50 in equivalent compute cost. The same fine-tuned model with external context injection achieved 100% accuracy (50/50) with no additional training — the only added artifact was a 640KB static JSON file

containing the per-objective decision guides.

Nested Learning interpretation. The fine-tuned model's internal associative memories operate at δ-band (pre-training weights) modified by the fine-tuning gradient (a slow, expensive update). The model learned verdict-generation capability — the *process* of assessing compliance — but lacked the *specific knowledge* of what each CMMC objective requires and what specific Graph API evidence proves compliance.

External context injection provides this knowledge at θ-band — persistent institutional knowledge injected at inference time. The fine-tuned model supplies the reasoning pattern; the external memory supplies the domain facts. This division mirrors NL's observation that different frequency bands serve different functions: low-frequency learning captures general patterns, while higher-frequency mechanisms capture specific, context-dependent information.

The "sweet spot" phenomenon is also interpretable through NL: the initial fine-tuning rounds trained the δ-band associative memory to an effective local optimum for the assessment task. Additional data introduced conflicting gradients (surprise signals from diverse examples) that disrupted the established frequency balance without improving the overall compression quality. The external memory approach sidesteps this entirely by providing domain knowledge through a different frequency band rather than forcing it into the model's internal weights.

4.4 Bottery: Multi-Agent Nested Learning

Bottery is a container-based agent orchestration platform where each agent is a self-contained nested learning system with its own multi-timescale memory hierarchy:

```
agent-directory/
├── purpose.md          (δ — foundational identity, changes rarely)
├── persona.md          (δ — behavioral characteristics)
├── drives.md           (θ — standing operational principles)
├── CLAUDE.md           (θ — project-level instructions)
├── memory/
│   ├── wiki/           (α — accumulated knowledge entries)
│   ├── experiences/    (β — episodic logs of significant events)
│   ├── models/         (θ — mental models and frameworks)
│   ├── mistakes/       (α — dedicated surprise-signal channel)
│   └── working_summary.md  (β — current state)
├── inbox.md            (γ — incoming messages, ephemeral)
└── tasks/              (β — active work items)
```

Each agent accumulates knowledge independently through its own memory hierarchy. Cross-agent knowledge sharing occurs through a shared memory server (Nellie), which provides "wide" knowledge across all agents, while each agent's local memory provides "deep" knowledge for its specific domain.

The multi-agent system as nested optimization. In NL's framework, a multi-

agent system is itself a nested learning system: each agent is a learning module with its own context flow (individual memory hierarchy), and the coordination layer (inter-agent communication via the Beer Can message bus) is the outer optimization loop. The coordinator does not modify individual agents' weights or memories; it routes tasks based on agent capabilities and accumulates system-level observations about which agent performs best on which task types — an external analog of NL's self-modifying learning module.

Purpose drift detection. Each agent periodically compares its recent activity against its `purpose.md` and `drives.md` files. Deviation triggers a self-assessment. This is confidence decay in action: the agent's operational behavior is measured against its δ-band identity, and significant divergence (low confidence in purpose alignment) triggers corrective action at higher frequency bands.

4.5 AI Gateway: Organizational-Scale External Memory

The AI Gateway is a five-layer open-source proxy stack that sits between every user in an organization and every LLM provider, providing transparent audit capture without modifying user workflow.

Relevance to external nested learning. Every interaction flowing through the gateway is a data point in the organizational γ-band. The gateway's audit trail feeds the Praxis flywheel: aggregate interaction patterns (what questions are asked, which models are used, where quality issues arise) consolidate into organizational θ-band institutional knowledge. Over time, the gateway learns the organization's AI usage patterns and can proactively improve prompt quality, model routing, and safety guardrails without any individual user changing their behavior.

The key architectural decision is *transparent passthrough*: users set a single environment variable (`ANTHROPIC_BASE_URL`) to route requests through the gateway, and their existing authentication and workflow remain unchanged. The gateway captures everything but changes nothing about the user experience. This is the organizational equivalent of hook-based integration — the system is augmented at a layer the user (and the model) does not perceive.

Compliance architecture. In regulated environments, the gateway enforces a clean data boundary: sensitive interaction data stays within the organization's audit trail (γ-β bands), while only anonymized, pattern-level knowledge feeds the Praxis flywheel (α-θ bands). Customer data never touches training pipelines. The separation is architectural, not policy-based — different systems handling different frequency bands, with no pathway between them.

---

5. Discussion

5.1 Why External Beats Internal for Production Systems

The case for external nested learning over internal continual learning approaches rests on four properties:

Auditability. In regulated environments (CMMC, FedRAMP, HIPAA, SOX), organizations must demonstrate what their AI systems know and how they arrived at their answers. A fine-tuned model is a black box — the knowledge is distributed across billions of parameters, and no audit can determine what specific training example caused a specific output. The harness layer, by contrast, is fully inspectable: every injected context item has a provenance trail, every lesson has an author and timestamp, every correction is logged. An auditor can reconstruct exactly what the model "knew" at the time of any given response.

Replaceability. External memory is model-agnostic. The same Nellie lessons, Praxis layers, and SallyPort context maps work with Claude, GPT, Gemma, and local models. When a new model version is released (or an existing version is deprecated), the accumulated knowledge transfers instantly — change the model, keep the memory. Fine-tuned weights, by contrast, are locked to a specific model architecture and often a specific version. The harness is the durable asset; the model is a replaceable component.

Economics. Fine-tuning costs scale with model size: each round requires GPU hours proportional to parameter count. External memory costs scale with knowledge volume, independent of model size. A 640KB context map costs nothing to "deploy" to any model. This asymmetry becomes more pronounced as models grow larger: fine-tuning a 100B+ parameter model is prohibitively expensive for most organizations, while external context injection remains constant-cost.

Speed. A new lesson is available to all agents within seconds of creation. A new fine-tuning round requires hours to days of training, evaluation, and deployment. In fast-moving domains (active security incidents, evolving compliance requirements, live customer engagements), the ability to update the system's knowledge in seconds rather than days is operationally critical.

5.2 The Discard Pile: External Ensemble Selection

The SallyPort fine-tuning process produced multiple model checkpoints (v1-fix, v2, v3), each with different strengths. While v3 achieved the highest aggregate accuracy, v2 outperformed v3 on specific CMMC control families. In a traditional fine-tuning pipeline, inferior checkpoints are discarded.

We propose retaining the top-N checkpoints and using the harness to route queries to the best-performing checkpoint per domain — an external ensemble selection mechanism. The harness tracks per-checkpoint accuracy by control family (or more generally, by query category), and routes each new query to the checkpoint with the highest historical accuracy for that category.

In NL's framework, each checkpoint represents a different local optimum of the δ-band associative memory — each has internalized a slightly different compression of the training data. The harness provides the outer optimization that selects the best inner optimization per query. This is functionally equivalent to NL's description of nested optimization processes, but implemented externally where

the selection logic is inspectable and updatable without retraining.

5.3 Limitations

Latency overhead. Context injection adds latency at every turn — memory search, retrieval, and prompt augmentation occur before the model receives the prompt. In our systems this overhead is 100–500ms, acceptable for interactive use but potentially problematic for latency-critical applications.

Token budget constraints. Models have finite context windows. Every token spent on injected context is a token unavailable for the conversation itself. Hierarchical summarization (§3.2) mitigates this, but there is an inherent tension between memory richness and conversational bandwidth. Current frontier models with 200K+ token windows significantly alleviate this constraint compared to the 4K-8K windows of two years ago.

Reasoning quality floor. The harness can inform the model but cannot upgrade its fundamental reasoning capabilities. A model that lacks the ability to reason about compliance concepts will not produce correct assessments regardless of how much compliance knowledge is injected. External nested learning improves the model's effective capability but is bounded by its base reasoning quality.

Empirical, not formal. We provide empirical evidence across five production systems but no formal convergence proof for external nested learning. Whether the consolidation pipeline (§3.3) converges to optimal knowledge representations under general conditions remains an open theoretical question.

5.4 Connection to the Periodic Table of Data Manifolds

Morris and Claude (2026) show that data types cluster into families with predictable transfer properties, organized by intrinsic dimensionality and compressibility — a "periodic table" of data manifolds where structural properties predict scaling behavior.

The harness-layer architecture enables a form of cross-modal transfer without retraining: institutional knowledge about one data type (e.g., compliance text) can inform handling of another (e.g., structured API response data) through the shared memory hierarchy. A lesson learned from text-based compliance documents at α-band can improve the model's handling of JSON evidence payloads because the harness injects the lesson as context regardless of the input modality.

The "substrate coupling" axis proposed by Morris (the degree to which the processing architecture changes the effective dimensionality of the data) maps to the model-harness coupling question: how much does the harness change the effective capability of the model? Our SallyPort results suggest that for domain-specific tasks, the answer is "substantially" — from 84% to 100% on the same benchmark, representing a phase transition from competent to expert-level performance driven entirely by external memory.

5.5 Toward a Self-Describing Harness

If the harness provides external nested learning for the model, what provides it for the harness itself? The harness's own consolidation logic, retrieval strategies, and confidence calibration are themselves knowledge that could benefit from continual improvement.

The Vector-Relational Data Model (VRDM) offers a path: the harness's architecture is expressed as a VRDM model — a self-describing data structure where concepts, relationships, and services are defined using the same primitives. The VRDM meta-model is reflexive: it describes itself using its own type system. This means the harness can inspect, validate, and modify its own structure using the same tools it uses to manage domain knowledge.

This reflexive property is NL's "Self-Modifying Learning Module" at the systems level. The harness learns how to be a better harness — which retrieval strategies work best for which query types, which confidence thresholds produce optimal context injection, which consolidation intervals balance freshness against stability. The harness modifies its own learning rules, not just its memory contents.

---

6. Related Work

Nested Learning. Behrouz et al. (2025) provide the theoretical foundation for this work. Their Continuum Memory System, multi-timescale update framework, and Self-Modifying Learning Module describe internally what our architecture implements externally. We view our work as an applied complement: NL provides the "why" (continual learning requires multi-frequency memory); we provide the "how" for deployed, frozen models.

Context Engineering. Hua et al. (2025) survey context engineering across four eras, identify the need for lifelong context preservation, and envision a "semantic operating system for context." They cite Claude Code's subagent isolation as an exemplar of context management. Our architecture implements their vision with the additional constraint of multi-timescale consolidation and surprise-driven refinement.

Agent Memory Systems. memU (NevaMind-AI, 2025) provides a three-layer memory hierarchy (Resource → Item → Category) for proactive agents, achieving 92% accuracy on the Locomo benchmark. Their Resource/Item/Category layers parallel our γ/α-β/θ bands. The key difference is integration approach: memU operates inside the agent's inference loop, while our architecture operates outside the model entirely, at the harness layer.

The Hindsight framework (vectorize-io, 2025) implements biomimetic retain/recall/reflect cycles with three memory types. Their reflect operation — consolidating experiences into more abstract knowledge — corresponds to our α→θ consolidation pipeline.

The agentic-stack project (codejunkie99, 2026) introduces an “auto-dream” mechanism: nightly processing that clusters recurring patterns from episodic memory into candidate lessons, which a host agent reviews and graduates or rejects. This automated consolidation is a specific implementation of our β→α transition.

Confidence and Decay. KNDL (Knowledge Node Description Language) provides formal support for confidence-native assertions with temporal decay and bitemporal tracking. Our confidence-weighted knowledge (§3.5) draws on KNDL’s framework for treating confidence as a first-class property of stored knowledge.

Knowledge Compaction. The Beads framework (gastownhall, 2025) provides atomic claims with hash-based identifiers and memory compaction — mechanisms for managing growing knowledge stores without unbounded storage growth. Our consolidation pipeline includes compaction at each frequency transition, where multiple high-frequency items are compressed into fewer low-frequency items.

LLM Wiki. Karpathy (2026) demonstrates file-based persistent knowledge management for LLMs using schema-driven, incrementally-cached wiki entries. This approach corresponds to our α-θ band storage and validates the file-based, Markdown-first approach we adopt for human-inspectable knowledge stores.

---

7. Conclusion

Behrouz et al. (2025) prove that neural networks are nested hierarchies of associative memories with multi-timescale context flow, and that continual learning requires multi-frequency memory consolidation. We show that when deployment freezes these internal dynamics, a harness layer can restore them externally — providing the multi-frequency memory consolidation, surprise-driven refinement, and confidence-weighted knowledge that deployed models lack.

The brain does not solve continual learning by retraining the cortex. It builds a separate memory system — the hippocampus — that consolidates experience across timescales and injects consolidated knowledge when needed. We built the same architecture for LLMs: the harness is the hippocampus, the model is the cortex, and the separation is what makes the system both continually learning and production-safe.

Five production systems demonstrate that this approach works across domains — compliance assessment (SallyPort, CMMC-Buddy), code assistance (Nellie), multi-agent coordination (Bottery), expert system construction (Praxis), and enterprise AI governance (AI Gateway) — with measurable improvements in accuracy, efficiency, and knowledge retention, at zero retraining cost. Our strongest result — 100% accuracy on a compliance benchmark where fine-tuning alone plateaus at 84% — suggests that external memory at the right timescale can surpass internal learning for domain-specific tasks.

The implication for the field is architectural: the pursuit of continual learning may

not require modifying model weights at all. If the missing capability is multi-timescale memory consolidation, and if that capability can be provided externally, then the model can remain frozen — auditable, replaceable, and compliance-safe — while the harness handles learning. The model reasons. The harness remembers. And the separation is the point.

References

Achiam, J., et al. (2023). GPT-4 Technical Report. *arXiv:2303.08774*.

Akyürek, E., et al. (2024). In-context learning with long-context models. *arXiv:2405.00200*.

Behrouz, A., Razaviyayn, M., Zhong, P., & Mirrokni, V. (2025). Nested Learning: The Illusion of Deep Learning Architecture. *NeurIPS 2025*. arXiv:2512.24695.

Brown, T., et al. (2020). Language Models are Few-Shot Learners. *NeurIPS 2020*.

Comanici, G., et al. (2025). Claude: A Family of Capable, Aligned AI Assistants. *Anthropic Technical Report*.

Eyuboglu, S., et al. (2025). Continual learning without catastrophic forgetting. *arXiv:2501.xxxxx*.

Foster, D. J., & Wilson, M. A. (2006). Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084), 680–683.

Hua, Y., et al. (2025). Context Engineering 2.0: The Context of Context Engineering. *arXiv:2510.26493*.

Karpathy, A. (2026). LLM Wiki: Persistent Knowledge Management for Language Models. *Blog post*.

Morris, M., & Claude (2026). Toward a Periodic Table of Data Manifolds. *Preprint*.

Scoville, W. B., & Milner, B. (1957). Loss of recent memory after bilateral hippocampal lesions. *Journal of Neurology, Neurosurgery & Psychiatry*, 20(1), 11–21.