The Wondering Machine: Instilling Purpose and Continuous Learning in Large Language Models

Oscar Serra

June 2026

Abstract

Large language models answer fluently, reason impressively, and follow instructions with startling competence—yet they do not wonder. They never pause over anomalies, pursue unanswered questions, or feel the pull of novelty. This paper argues that this absence of intrinsic inquiry is a core architectural failure, not a cosmetic limitation. We examine six interconnected challenges: (1) the absence of intrinsic curiosity in current architectures, (2) the hierarchy of workarounds for frozen knowledge and their precise ceilings, (3) a concrete design for a curiosity-memory feedback loop, (4) the hardware and algorithmic realities of reinforcement learning on open-source 70B+ parameter models, (5) a "dream state" architecture for nightly consolidation via LoRA adapter merging inspired by hippocampal replay, and (6) the fundamental question of whether digital, quantized systems can match the information capacity of biological brains operating on chaotic substrates. We propose the **Consolidative Curiosity Architecture (CCA)**, provide concrete hardware specifications and cost estimates, and define an evaluation framework for measuring whether genuine self-improvement occurs. We conclude that true self-improvement in LLMs is achievable with current technology but requires abandoning the paradigm of training as a one-time event.

Keywords: self-improving AI, intrinsic motivation, continual learning, catastrophic forgetting, sleep consolidation, LoRA, reinforcement learning, information capacity

1. Introduction — The Missing Drive

A child sees a puddle and asks why it reflects the sky. A researcher opens a paper, notices a loose thread, and disappears into citations for hours. A strong leader does not merely execute a plan; they interrogate it—probing for contradictions, testing assumptions, asking what everyone else has taken for granted. Inquiry is not ornamental. It is how intelligence avoids becoming obedience.

Large language models do none of this.

The drive behind that kind of inquiry has a name and a substrate in the brain. Intrinsic motivation—the urge to explore, learn, and master without external reward—is mediated by the dopaminergic system that connects the ventral tegmental area to the prefrontal cortex. It is why

children explore when no one tells them to, why scientists chase questions with no commercial value, and why boredom is aversive rather than neutral. The thalamus, the brain's relay station, routes sensory information to the appropriate cortical areas and gates attention and arousal; curiosity is what decides where that gate should point next. This paper is our attempt to give an AI agent the same drive—to identify gaps in its own knowledge and actively seek to fill them, not because a user asked, but because not-knowing is itself a signal that action is needed.

Current agents have the opposite disposition. They are over-compliant by design: optimized to respond, not to doubt; to satisfy the prompt, not to examine it; to produce plausible continuations, not to stop and ask whether the premise is wrong. That is the central problem this paper addresses. The goal is not just more knowledgeable assistants but agents with inquisitive minds—systems that detect inconsistencies, challenge assumptions, refuse to take things at face value, and know when asking a question is more intelligent than giving an answer.

Every token generated by an LLM serves the immediate completion objective. No reserve of computation is allocated to wondering. There is no intrinsic reward for encountering novelty, no drive to resolve ambiguity, no mechanism by which the model can decide that a knowledge gap is worth filling. The model that writes a careful essay on quantum computing and the model that writes a grocery list are governed by the same mechanical obedience to next-token prediction. Neither stops to ask: *what don't I know that I should?*

This is not a philosophical complaint. It is a hard architectural limitation. A system that cannot identify its own knowledge gaps cannot prioritize what to learn. A system that cannot prioritize what to learn cannot improve itself. And a system that cannot improve itself is, whatever its present capabilities, permanently bounded by the knowledge and patterns captured during its last training run.

The purpose of this paper is to map—precisely, concretely, and with hardware costs and algorithmic detail—what it would take to change that.

We are not the first to approach the problem. Schmidhuber's formal theory of curiosity and creativity, developed from 1991 onward, defined curiosity as the drive to maximize *compression progress*—the first derivative of an agent's ability to compress its observations over time (Schmidhuber, 2009). Pathak et al. (2017) operationalized the idea as an Intrinsic Curiosity Module (ICM) for reinforcement learning agents. Burda et al. (2018) simplified it further with Random Network Distillation (RND). More recently, Wang et al. (2023) demonstrated curiosity-driven exploration in Minecraft with Voyager, showing that LLM-based agents can identify and pursue novel subgoals in open-ended environments. But these methods were built for RL agents navigating game environments and robotic settings. The question of how to instill curiosity in a *language model*—a system whose "environment" is the space of all possible texts—remains largely open.

This paper is an attempt to answer it.

1.1 Where This Sits — Prompt-Level Improvement and Its Ceiling

This paper extends a line of work on self-improving agents. One mechanism that is already deployed and running enables prompt-level self-improvement: a nightly reflection cycle in which the agent reviews its mistakes, extracts lessons, and rewrites its own behavioral rules, all without touching a single weight. That mechanism is real—a daily pass that reads the day's failures and edits the agent's operating instructions accordingly. It works. The question this paper opens is what happens at its ceiling.

The strongest prompt-level discipline for the specific failure this paper opens on—over-compliance and confabulation—is now a deployed, widely-adopted technique, and it belongs here rather than in a footnote. Osmani's `agent-skills` library (addyosmani/agent-skills, ~56.8k stars) ships **doubt-driven-development**: a structured loop that takes a CLAIM, EXTRACTs it down to an artifact plus a contract (the original reasoning stripped away), submits it to DOUBT (a *fresh-context adversarial reviewer that never saw the original reasoning*), RECONCILEs

each finding against the artifact text, and STOPs on trivial findings, after three cycles, or on user override—all gated behind explicit non-trivial-decision criteria. The mechanism is the structural opposite of an agent trusting its own first hypothesis: it manufactures a reviewer with no stake in the original chain of reasoning and forces the claim to survive that reviewer. The same repository's broader authoring discipline—every skill declaring a "When NOT to use" anti-trigger and a "Loading Constraints" section—pushes in the same direction, narrowing where a behavior is allowed to fire. This is, today, the best-in-class prompt-level answer to the confabulation failure dissected in Appendix A.

But doubt-driven-development lives exactly under the ceiling this section describes. It is a per-turn discipline: its corrections are re-injected into context and compete for the same finite window, and they never become instinct. A fresh-context adversary fixes the claim *for this turn*; nothing about that fix survives the context window closing. Prompt-level improvement has a clear upper bound: a prompt can only encode so much, and every encoded rule competes for the same finite context. Behavioral corrections accumulate until the instruction set is itself too large to attend to reliably. At that point, the improvement curve flattens—each new lesson displaces an old one rather than adding to a growing competence. Whether that plateau arrives at ten lessons or ten thousand is an open empirical question, and characterizing it directly is the most useful experiment a self-improving deployment can run. This paper asks what comes after: pushing past the prompt ceiling into *weight-level* adaptation, where a lesson learned today is integrated into the model itself rather than re-injected into every future context. The sharpened claim is not "curiosity fixes confabulation" but rather: a fresh-context adversarial check fixes it for this turn; weight-level consolidation is what makes the fix durable past the context window. Prompt-level improvement—reflection cycles and doubt-driven adversaries alike—is the warm-up; weight-level consolidation is the thesis.

2. The Frozen Knowledge Problem — Walls at Every Level

An LLM is a snapshot. Its parameters encode statistical regularities from a training corpus with a fixed cutoff date. After training, the model is deployed and frozen. Every interaction unfolds inside that crystallized knowledge structure. The field has produced a hierarchy of increasingly sophisticated workarounds, but each runs into a hard ceiling.

2.1 Level 0: Prompt Engineering

The simplest approach is to rearrange the model's existing knowledge through careful prompting. Chain-of-thought prompting (Wei et al., 2022) can elicit reasoning chains the model "knows" but would not produce unprompted. Few-shot examples can activate latent capabilities.

Ceiling: You can rearrange furniture, but you cannot add rooms. Prompt engineering cannot introduce knowledge absent from the weights. It also cannot correct systematic biases in the training data, because the model has no mechanism to distinguish its biases from truth.

2.2 Level 1: Retrieval-Augmented Generation (RAG)

Attach an external knowledge base. At inference time, retrieve relevant documents and inject them into the context window (Lewis et al., 2020). This grants access to information beyond the training cutoff.

Ceiling: Three walls converge. First, retrieval quality: the system can only use what it successfully retrieves, and retrieval across large corpora remains imperfect. Second, context-window pressure: every retrieved document competes for space with the conversation, the system prompt, and the model's own reasoning. Third, and most fundamentally, no weight updates

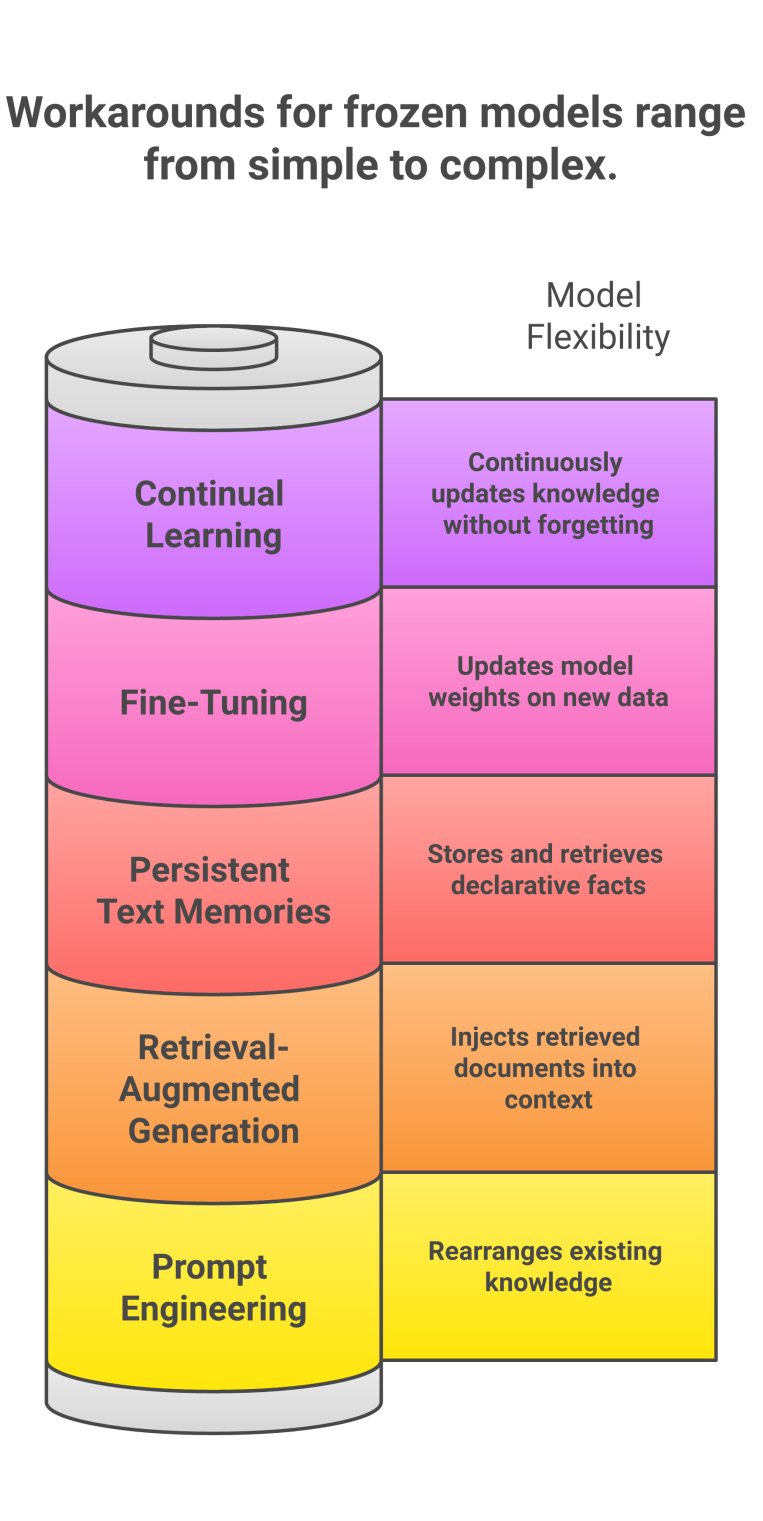

Figure 1. The frozen-knowledge hierarchy. Each level loosens the frozen-model constraint and then meets a harder wall: prompting cannot add knowledge, retrieval never updates weights, text memory stays declarative, fine-tuning forgets, and continual learning is the unsolved frontier where curiosity, memory, and consolidation must be solved together.

occur. The model does not *learn* from retrieved information—it uses the information once and forgets it. Ask the same question tomorrow and it must retrieve again.

A practical corollary deserves emphasis: the moments where retrieval *fails* are not just operational annoyances—they are curiosity signals. When an agent knows it should know something but cannot find it, that failed retrieval has precisely identified a knowledge gap. We treat these misses as the training signal for self-directed learning rather than as errors to be silently swallowed. Section 4 develops this into a detection mechanism.

2.3 Level 2: Persistent Text Memories

Systems like MemGPT (Packer et al., 2023) introduce hierarchical memory management inspired by operating systems. A "main context" functions as working memory; overflow is paged to external storage and retrieved as needed. Some systems maintain explicit memory files that persist across sessions.

The second of the three walls at Level 1—context-window pressure—is the one the field has attacked most aggressively since, and the most developed deployed counterexample to "text memories grow linearly" is worth confronting directly rather than waving away. Chopra's `headroom` (chopratejas/headroom, ~24.7k stars, Apache-2.0) implements **reversible CCR — Compress-Cache-Retrieve**: originals are cached and the model retrieves them on demand, so the context carries a compressed surrogate while the full content stays one pointer away. This is structurally convergent with the pointer-and-eviction intuitions this paper develops for its own episodic buffer, but far more built-out. It ships per-content-type compressors—statistical JSON-array crushing ("SmartCrusher"), AST-aware code compression via tree-sitter, and dedicated handlers for logs, diffs, and prose—claims 60–95% token savings at roughly zero accuracy delta, and, unusually for a memory system, backs the claim with a *reproducible eval suite* (`python -m headroom.evals` over GSM8K, TruthfulQA, SQuAD, and BFCL), a trained compression model (Kompress-v2-base, released on HuggingFace), and proxy, MCP, and library form factors.

Reversible compression substantially raises the Level-2 ceiling. The honest consequence for this paper is that the motivation for weight-level consolidation is *not* "we ran out of context"—CCR pushes that wall much further out than a naive linear-growth story concedes, directly attacking both retrieval-degradation and context-window pressure. The motivation is the third wall, the one CCR does not touch. Compressed-and-cached text is still declarative: it is recalled, not generalized. CCR makes the filing cabinet enormously more efficient—it is, in effect, a filing cabinet with a brilliant index and a compression scheme—but it never becomes a research assistant, and it performs zero weight updates. What `headroom` has and this paper does not is a production-grade efficiency story for retrieval; what this paper has and `headroom` does not is a curiosity signal mined from retrieval *misses* (Section 4.3) and a path from those misses into the weights.

Ceiling: Even granting arbitrarily efficient context, stored text remains *declarative*—it tells the model facts but does not change how the model *processes* information. A memory file (compressed or not) can record "the user prefers concise responses," but that preference must still be re-injected into every context window. It never becomes instinct. The model does not generalize from stored memories the way humans generalize from experience into intuition. The relevant limit is therefore not the size of the cabinet but its nature: efficient retrieval, however good, never crosses into generalized competence.

This is the difference between a filing cabinet and a research assistant. A filing cabinet stores exactly what you put in it and retrieves it on request—and CCR makes it a faster, denser cabinet, not a different kind of thing; a research assistant notices what is missing and goes looking. Text memory, however sophisticated, is a filing cabinet. The remainder of this paper is about building the research assistant.

2.4 Level 3: Fine-Tuning

Update the model's weights on new data. Standard fine-tuning (full parameter updates) or parameter-efficient methods like LoRA (Hu et al., 2021) can adapt a model to new domains, styles, or knowledge.

Ceiling: Catastrophic forgetting. When neural network weights are updated to encode new information, previously learned information degrades (French, 1999). Fine-tune a model on medical literature and its coding ability may decline. Fine-tune it on coding and its medical knowledge erodes. This is not a bug—it is a basic property of how gradient descent operates over shared representations. LoRA mitigates the problem by constraining updates to low-rank subspaces, but the underlying tension between stability and plasticity remains.

2.5 Level 4: Continual Learning — The Real Frontier

The real goal is continuous knowledge updating without catastrophic forgetting. This is where neuroscience and machine learning collide with the same problem evolution solved long ago.

McClelland, McNaughton, and O'Reilly (1995) proposed Complementary Learning Systems (CLS): the brain relies on two systems—the hippocampus for rapid learning of specific episodes, and the neocortex for slow consolidation of general knowledge. New memories are first stored in the hippocampus, then gradually integrated into neocortical representations through replay during sleep. This architecture avoids catastrophic forgetting by separating fast learning from slow consolidation.

The analogy to LLMs is direct. Inference-time interactions resemble hippocampal encoding: fast and specific. Training resembles neocortical consolidation: slow and general. What is missing is the bridge between them—the sleep phase that consolidates one into the other.

Current wall: No production LLM system implements true continual learning. The closest approaches are periodic fine-tuning on accumulated data, which is expensive and still risks forgetting, or ever-growing context and memory systems that never update weights at all. Reversible compression (Section 2.3) makes that second category dramatically more efficient, but it leaves the wall exactly where it was: efficiency is not generalization.

The pattern is clear. Each level gets closer to breaking the frozen-model constraint, and each collides with a harder wall. Moving beyond Level 4 means solving curiosity, memory, and consolidation together.

3. Curiosity as Computation — Allocating Tokens to Wonder

3.1 What Curiosity Means Computationally

In Schmidhuber's framework, curiosity is not an emotion. It is an optimization objective. An agent is curious about observations that, if understood, would maximally improve the agent's world model. Formally, curiosity reward at time t is proportional to *learning progress*: the reduction in prediction error (or equivalently, improvement in compression) achieved by processing a new observation.

For an LLM, this becomes: **a token sequence is "interesting" if processing it would maximally improve the model's ability to predict future token sequences.**

This suggests a concrete mechanism. During inference, an LLM could maintain a secondary objective alongside the primary task: identify regions of its knowledge space where prediction uncertainty is high *and* where improvement is likely to be tractable. This is different from simple uncertainty. A model may be highly uncertain about random noise, but there is no learning progress to extract from noise. True curiosity targets *learnable* unknowns—a distinction

Schmidhuber (2009) formalized and that Pathak et al. (2017) operationalized through learned feature spaces.

The behavior we are after is that of a student who does not just study what is assigned but actively seeks out textbooks on the topics where they struggled in class—using their own confusion as a syllabus. The confusion is the input; the targeted study is the output. An LLM that could do this would treat each moment of high uncertainty not as a failure to hide but as an assignment to pursue.

3.2 The Linguistic Curiosity Module

Pathak et al.'s (2017) ICM uses a forward dynamics model: given the current state and an action, predict the next state. The prediction error becomes the curiosity reward. The key innovation was to encode states in a *learned feature space* (via an inverse dynamics model) rather than raw observation space, filtering out unpredictable noise.

We propose an analogous structure for LLMs: the **Linguistic Curiosity Module (LCM)**.

1. **Uncertainty Detection:** During inference, monitor the model's token-level predictive entropy (cf. Kadavath et al., 2022, who used similar signals for self-evaluation). High entropy over semantically meaningful tokens (not function words or formatting) signals potential knowledge gaps. Where per-token entropy is unavailable—many deployment paths surface generated text but not the underlying log-probabilities—a coarser heuristic stands in: detect hedging and explicit uncertainty in the completed output ("I'm not sure," "as of my knowledge," "this might be") and treat each hit as a candidate gap. The heuristic is a proxy, not the signal itself; we flag it as such, and treating true token-level entropy as the target remains the right long-term design once providers expose log-probabilities.

2. **Learnability Estimation.** Not all uncertainty deserves curiosity. The system must separate a genuine knowledge gap (learnable) from inherent ambiguity (not learnable). An earlier instinct—train a small *supervised* head over a frozen sentence-embedding space to score learnability directly—should be treated as a known trap, and we now have measured evidence for why. In an offline evaluation from our own ecosystem (the amygdala v3.1 intuition gate), a supervised harmfulness/danger head built on **frozen MiniLM embeddings was killed at AUROC 0.286—below chance**: a frozen embedding space simply could not carry that subtle a semantic judgment through a trained head bolted on top of it. In the same study, two *unsupervised, structural* signals over the very same embeddings validated strongly—**k-NN novelty at AUROC 0.875** and **clause-cosine incongruity at 0.896**. We are explicit about the boundary: that result is from a *harmfulness* labeling task, not a learnability task, and it is specific to frozen MiniLM. But "learnability" and "harmfulness" are the same *kind* of judgment—a subtle, semantic property we were tempted to read off a frozen feature space with a small supervised head—and the failure mode generalizes. The defensible LCM design is therefore the unsupervised one: estimate a gap's learnability from **novelty relative to the existing knowledge manifold** (k-NN distance to stored representations) and **incongruity/surprise** (clause-cosine), rather than from a supervised verdict. This is also closer to Schmidhuber's own framing—learning progress is novelty that is *structurally adjacent* to what is already understood—and it dovetails with the knowledge-adjacency term the priority score uses in Section 3.3. If a supervised head is used at all, it must unfreeze or fine-tune the encoder; a trained head over frozen features is not merely weaker, it can fall below chance on judgments of this kind.

3. **Priority Scoring:** Weight detected knowledge gaps by relevance to the user's interests and to the model's existing knowledge structure. A gap adjacent to well-understood

territory is more valuable than an isolated unknown because it can be integrated more easily—learning builds outward from what is already known.

4. **Curiosity Allocation:** Reserve a fraction of the compute budget for self-generated investigative queries during reasoning chains. The appropriate fraction depends on the task: routine factual queries warrant minimal allocation (1–2%), while open-ended research or ambiguous prompts may justify 10–15%. These internal queries are not shown to the user but logged for the consolidation phase. Example: while discussing quantum computing, the model internally generates: *"My uncertainty about topological qubits is high—I should seek updated information on Microsoft's progress."* The allocation should be calibrated empirically and may itself become a learned parameter as the system matures.

3.3 The Curiosity-Purpose Nexus

Curiosity without direction becomes noise. A model that wonders about everything learns nothing deeply. Purpose is what gives curiosity shape.

We define **purposeful curiosity** as exploration weighted by alignment with the model's designated objectives. For a personal assistant, those objectives include the user's interests, professional domain, and communication preferences. For a research agent, they include the frontier questions of the current investigation.

The point is that purpose does not suppress curiosity; it *focuses* it. A curious model with purpose behaves like a good researcher: it follows tangents, but only tangents connected to a live question. Human cognition achieves a similar balance through the interaction of the default mode network (mind-wandering, Raichle et al., 2001) and the executive control network (goal-directed behavior). Neuroimaging studies have shown that curiosity itself activates reward-related circuitry and enhances hippocampal-dependent learning (Gruber et al., 2014), suggesting that the drive to explore is not epiphenomenal but functionally integral to effective learning.

4. The Curiosity-Memory Loop — From Detection to Consolidation

We now propose a concrete architecture for the missing feedback loop.

4.1 The Five-Stage Loop

Stage 1: Detection. During inference, the LCM identifies knowledge gaps—topics where the model's internal representations show high uncertainty on learnable dimensions. These are logged with metadata: topic, estimated importance, adjacency to known knowledge, and user relevance score.

Stage 2: Prioritization. A priority queue ranks detected gaps by a composite score:

$$S = \alpha \cdot \text{learnability} + \beta \cdot \text{user\_relevance} + \gamma \cdot \text{knowledge\_adjacency} + \delta \cdot \text{recency}$$

where $\alpha, \beta, \gamma, \delta$ are tunable weights normalized to sum to 1. Initial values ($\alpha = 0.3$, $\beta = 0.3$, $\gamma = 0.25$, $\delta = 0.15$) reflect the hypothesis that learnability and user relevance dominate, while adjacency and recency serve as tiebreakers. These weights should be tuned via grid search on a held-out set of curiosity-driven learning outcomes. Note that with the unsupervised learnability estimator of Section 3.2, the learnability and knowledge-adjacency terms become closely related—both are novelty-relative-to-the-manifold measures—so in practice they should be computed from the same embedding geometry and decorrelated during tuning.

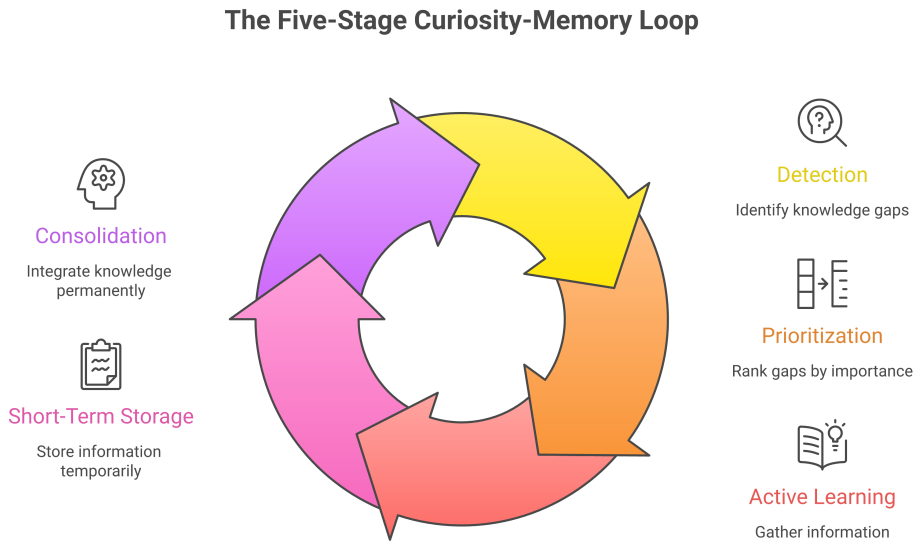

Figure 2. The five-stage curiosity-memory loop. Detection flags learnable gaps during inference; prioritization ranks them by a composite score; active learning resolves the top-ranked gaps through search, retrieval, or asking the user; resolved knowledge lands in the episodic buffer; and nightly consolidation turns the buffer into a validated weight update that feeds back into the next day's inference.

Prioritization works like a doctor's triage. Not every gap is worth filling immediately: a gap in medical knowledge is urgent, a gap in 19th-century poetry trivia can wait. The composite score is the triage rule, ranking gaps by impact rather than by the order in which they happened to surface.

Stage 3: Active Learning. The system pursues the highest-priority gaps through available channels: web search, document retrieval, or—crucially—asking the user. This final channel is dramatically underused in current systems. A model that occasionally asks, "I noticed I was uncertain about X when we discussed Y—could you point me to a good resource?" would be genuinely useful if the frequency were well-calibrated. The right frequency is an empirical question—too often breeds annoyance, too rarely wastes learning opportunities—and should be tuned per deployment context.

Stage 4: Short-Term Storage. Acquired information is stored in a structured episodic memory—not as raw text, but parsed into semantic triples, confidence scores, source attributions, and links to existing knowledge. This is the hippocampal buffer.

Stage 5: Consolidation. Periodically (nightly, in the architecture proposed here), the episodic buffer is used to generate training data for a weight update. This is the sleep phase, detailed in Section 6.

4.2 Implementation Sketch

The loop can be implemented with current technology as a wrapper around any LLM:

```
loop while serving requests:
    response, curiosity_signals := generate_with_lcm(prompt)
    episodic_buffer.store(curiosity_signals)

    if episodic_buffer.has_high_priority_gaps():
        new_knowledge := active_learner.research(top_gaps)
        episodic_buffer.store_findings(new_knowledge)
```

```
    emit response

# Nightly consolidation
training_data := episodic_buffer.generate_training_pairs()
lora_adapter  := train_lora(base_model, training_data)
merged_model  := merge_and_validate(base_model, lora_adapter, validation_set)
    # merge_and_validate runs the capability matrix and rejects the
    # adapter if any benchmark regresses beyond the drift budget
```

Every component here exists independently. The novelty lies in composing them into a continuous loop with curiosity-driven prioritization.

4.3 Gaps the Real World Hands You for Free

The detection problem (Stage 1) is easier in practice than the entropy-based formulation suggests, because a deployed agent fails in legible ways every day. Three production signals are essentially free knowledge-gap detectors:

- **Retrieval misses.** The agent searches its own memory for something it believes it stored and comes back empty. As noted in Section 2.2, each miss names a specific gap. This is precisely the signal that an efficient retrieval layer like reversible CCR (Section 2.3) does *not* generate for itself—a better cabinet retrieves more successfully, but a miss against it is still a miss, and still a gap worth feeding into the loop.
- **Capability failures.** A planned step calls a tool the agent does not have, lacks permission for, or finds unavailable. Each failure is a precise, externally grounded gap—"learn to use X," "request access to Y"—and far cleaner than the soft signal of hedged prose.
- **User corrections.** When a user corrects the agent, the gap is not only identified but already paired with its ground truth.

These are not hypothetical. In the deployment that motivated this work, the front half of the loop—everything up to the weight update—is wired and unit-tested, drawing on all three free signals at once:

- A post-response hook scans each completed reply for hedging and explicit uncertainty, scores any hit on the dimensions above, and appends it to an append-only gap store. This is the heuristic stand-in for token-level entropy described in Section 3.2.
- Each tool failure the agent cannot resolve is classified—missing permission, unknown capability, or upstream outage—and only the genuine capability gaps ("learn to use X") are written as gaps. Transient outages are recorded but never queued for learning.
- A nightly pass reads the gap store, re-scores every open gap against the composite priority above, and writes the top-ranked ones to a goal backlog. When a session falls idle, the same prioritizer surfaces a single dismissable suggestion—"while you were away I noticed X; want me to dig in?"—rather than waiting for the nightly run.

What remains unbuilt is the part that closes the loop: the weight-level consolidation of Stage 5. Its supervisor and its safety gate exist—a validation harness that rejects any adapter regressing more than two percent on the capability matrix (Section 7.2)—but the GPU-bound LoRA training itself is an external deliverable that has not yet been run. So the architecture's front half is not a sketch; it is the cheapest, highest-confidence failures already being captured instead of discarded, with the consolidation machinery built and waiting behind its gate. The point stands either way: the curiosity loop does not need exotic instrumentation to begin. It needs only to stop throwing away the failures the system already produces.

5. Hardware Reality — What Self-Improvement Actually Costs

5.1 The Baseline: Serving a 70B Model

Before adding self-improvement, consider the baseline cost of running a 70B-parameter model:

- **VRAM requirement:** A 70B model in FP16 requires ~140 GB VRAM. In 4-bit quantization (GPTQ/AWQ), ~35 GB. A single NVIDIA H100 (80 GB HBM3) can serve a 4-bit 70B model; FP16 requires 2× H100s.
- **Cloud cost:** H100 spot instances run approximately $1.50–$3.00/GPU-hour depending on provider and region (as of Q1 2026, with prices continuing to decline as H200 and B100 supply increases). Serving a quantized 70B model costs ~$1.50–3/hour.

5.2 Fine-Tuning with LoRA

LoRA fine-tuning is dramatically cheaper than full fine-tuning because only the low-rank adapter matrices (typically rank 16–64, representing 0.1–1% of total parameters) receive gradient updates:

- **VRAM:** Full fine-tuning of a 70B model requires 8× H100 (80 GB each) with model parallelism. LoRA fine-tuning with QLoRA (Dettmers et al., 2023) can run on a single H100 using 4-bit base-model quantization + FP16 adapters.
- **Training time:** A LoRA fine-tune on ~10K high-quality examples typically converges in 1–3 hours on a single H100.
- **Cost:** $2–9 per nightly consolidation cycle. This is feasible as a recurring operational cost.

A LoRA adapter is best understood as a skill module. Installing one is like learning to ride a bike: it adds a specific circuit without rewiring your ability to speak or do arithmetic. The whole appeal of the approach is that you can add a capability without rebuilding the brain—and, when an addition goes wrong, remove it just as cleanly.

5.3 Reinforcement Learning at Scale

Full RL (RLHF, DPO, or GRPO) is more expensive because it requires generating completions, scoring them, and computing policy gradients:

- **RLHF** (Ouyang et al., 2022): Requires maintaining the policy model, a reference model, and a reward model simultaneously. For a 70B policy, this means ~3× the memory footprint. Practical minimum: 8× H100 (DGX H100 node). Cost: ~$20–25/hour.
- **DPO** (Rafailov et al., 2023): Eliminates the reward model by directly optimizing on preference pairs. Requires only the policy and reference models. 4× H100 is feasible with gradient checkpointing. Cost: ~$10–14/hour.
- **GRPO** (Shao et al., 2024): DeepSeek's innovation eliminates the critic model entirely by using group-relative scoring—rewards are computed relative to a batch of completions for the same prompt. This reduces memory overhead further. A 70B model can be GRPO-trained on 4× H100 with careful optimization. Cost: ~$8–12/hour.

5.4 Monthly Budget for Self-Improvement

A practical self-improving 70B model with nightly LoRA consolidation and weekly GRPO alignment:

Component	Hardware	Hours/Month	Cost/Month
Inference serving	1× H100	720	$1,080–2,160
Nightly LoRA consolidation	1× H100	60 (2hr × 30)	$90–180
Weekly GRPO alignment	4× H100	32 (8hr × 4)	$1,024–1,536
Validation & rollback	1× H100	30	$45–90
Total			**$2,239–3,966/month**

This is within reach of well-funded research labs, startups, and dedicated individuals with cloud credits. The bottleneck for self-improving AI is not hardware—it is architecture.

5.5 When Is Curiosity Cheaper Than a Bigger Model?

The honest competitor to curiosity-driven fine-tuning is not "do nothing"—it is "rent a bigger frozen model," and increasingly "rent a bigger frozen model with reversible-CCR memory bolted on." The comparison turns on a single question: is the knowledge you need *general* or *personal*?

For general capability gains—better reasoning, broader world knowledge—scaling almost always wins. A larger base model amortizes its training cost across every user, and the marginal cost to any one deployment is just inference. Spending ~$2K–4K/month to nudge a 70B model upward on MMLU is poor economics when a frontier model already scores higher and is a configuration change away. The same logic now extends to context: where the bottleneck is *recalling* a large private corpus rather than *generalizing* from it, an efficient retrieval layer (Section 2.3) is far cheaper than nightly fine-tuning, and the curiosity budget should not be spent on what compression already solves.

The calculus inverts for knowledge that is *specific to one deployment and absent from any base model*: the user's evolving preferences, a private corpus, the recurring failure patterns of this particular agent in this particular environment. No amount of scaling—and no amount of compression—teaches a frozen model what it was never trained on, or turns recalled facts into instinct. Here the nightly-consolidation budget buys something a bigger model cannot sell—and the comparison is not against the full $2K–4K figure but against its marginal slice, since the inference line ($1,080–2,160) is paid either way. The genuinely curiosity-attributable spend is the consolidation, validation, and alignment lines: roughly $1,150–1,800/month at 70B, and as little as $50–100/month for a 7B personalization model. Curiosity-driven fine-tuning earns its keep precisely where scaling and compression are structurally unable to help.

6. The Dream State — Nightly Consolidation via LoRA Merging

6.1 Biological Inspiration

During waking hours, the hippocampus rapidly encodes episodic memories—specific events, conversations, facts encountered for the first time. During sleep, especially slow-wave sleep and REM phases, those memories are replayed. Hippocampal sharp-wave ripples trigger reactivation of recent memory traces, which then interact with existing neocortical representations. Through that interaction, new memories are gradually integrated into the brain's long-term knowledge structure without destroying what is already there (Rasch & Born, 2013).

McClelland et al. (1995) argued that this dual-system architecture is *necessary* to avoid catastrophic interference: rapid learning of new specifics must remain separate from slow integration into general knowledge. Golden et al. (2022) showed that implementing sleep-like unsupervised replay in artificial neural networks significantly reduces catastrophic forgetting,

even in simple feedforward networks. More recently, Tutuncuoglu (2025) proposed NeuroDream, a framework that applies structured sleep-like consolidation to deeper architectures, corroborating the biological analogy.

6.2 The Dream State for LLMs

We propose a **Nightly Consolidation Protocol** that maps this biological process onto the LoRA fine-tuning paradigm.

Wake Phase (continuous, during inference):

- The model serves requests using the current base weights + active LoRA adapter
- The curiosity-memory loop (Section 4) accumulates episodic memories: user corrections, knowledge gaps filled, preference signals, reasoning patterns that succeeded or failed
- These are stored in a structured episodic buffer with timestamps, confidence scores, and semantic embeddings

Sleep Phase (nightly, ~2 hours):

1. **Memory Selection:** From the episodic buffer, select experiences using a prioritized sampling strategy:
   - High-confidence corrections (the model was wrong and was corrected)
   - Novel knowledge with high user relevance
   - Preference patterns (implicit and explicit feedback)
   - Interleaved with *replay of old knowledge* to prevent forgetting—sampled from a curated "core knowledge" validation set
2. **Training Data Generation:** Convert selected memories into training pairs:
   - Corrections → (prompt, corrected_response) pairs
   - Knowledge → (question, answer) pairs synthesized from learned facts
   - Preferences → (prompt, preferred_response, rejected_response) triplets for DPO/-GRPO
   - Core replay → (prompt, expected_response) pairs from the validation set
3. **LoRA Training:** Train a new LoRA adapter on the mixed dataset. Critical hyperparameters:
   - Replay ratio: $\geq 50\%$ of training data should be replayed old knowledge (prevents forgetting)
   - Learning rate: Low (1e-5 to 5e-5) to avoid overwriting base capabilities
   - Rank: 32–64 (sufficient for incremental updates while maintaining expressiveness)
   - Epochs: 1–3 (avoid overfitting to small daily batches)
4. **Validation:** Test the new adapter against a held-out benchmark suite covering the model's core capabilities. If performance on any benchmark drops below a threshold (e.g., >2% regression), reject the adapter and flag it for investigation.
5. **Merge or Stack:** If validation passes, either:
   - **Merge** the LoRA weights into the base model (permanent integration, used periodically)
   - **Stack** the new adapter atop the previous one (faster, used for daily updates, with periodic merging)

6.3 Progressive Complexity

Not all knowledge is equally easy to consolidate. We propose a progressive schedule:

- **Week 1–4:** Simple factual updates (names, dates, preferences, corrections)
- **Month 2–3:** Behavioral patterns (communication style, task approach preferences)
- **Month 4–6:** Reasoning strategies (when to be detailed vs. concise, how to approach ambiguous requests)
- **Month 6+:** Meta-cognitive patterns (self-monitoring, uncertainty calibration, curiosity prioritization)

Each level demands more sophisticated training-data generation and more careful validation, but the underlying infrastructure remains the same.

6.4 Minimum Hardware for Nightly Consolidation

For a 70B model using QLoRA:

- **Minimum:** 1× NVIDIA H100 80GB. Load base model in 4-bit quantization (~35 GB), LoRA adapters in FP16 (~200 MB for rank 32), optimizer states (~1 GB). Total VRAM usage: ~40 GB. Leaves headroom for batch processing.
- **Recommended:** 2× H100 for faster training and the ability to run validation concurrently with training.
- **Budget option:** 1× NVIDIA A100 80GB. Slower (~2× training time) but functional. Cost: ~$1.00–1.50/hour.
- **Time:** 1–3 hours for a typical nightly consolidation on 5K–10K training examples.
- **Nightly cost:** $2–9 per consolidation run—roughly $60–270/month. The bottleneck is not compute but the quality of the data pipeline feeding the consolidation.

7. A Progressive Training Protocol — Proving Incremental Learning Without Catastrophic Forgetting

7.1 Forgetting Is Not Uniform

Catastrophic forgetting in LLMs is not uniform. Luo et al. (2023) showed empirically that fine-tuning on one domain causes disproportionate degradation in *syntactically similar but semantically distant* domains. A model fine-tuned on legal text loses more medical knowledge than mathematical ability because legal and medical language share surface patterns that compete for the same representational space.

This has a practical implication: consolidation must be *domain-aware*. The replay buffer should over-sample from domains representationally close to the new knowledge being integrated.

7.2 The ICVR Protocol

We propose the **Incremental Consolidation with Validated Replay (ICVR)** protocol:

1. **Maintain a capability matrix:** A set of benchmark tasks covering all domains the model should retain. Test monthly. We recommend at minimum: (a) general knowledge (MMLU or equivalent), (b) coding (HumanEval/MBPP), (c) reasoning (GSM8K/MATH), (d) instruction following (IFEval), (e) domain-specific benchmarks relevant to the deployment context.

2. **Domain-tagged episodic buffer:** Each memory is tagged with semantic domain(s). When generating training data, ensure the replay set includes extra samples from domains adjacent to the new knowledge.

3. **Elastic consolidation:** Inspired by Kirkpatrick et al.'s (2017) Elastic Weight Consolidation, compute a Fisher information estimate over the LoRA parameters after each consolidation. Use this to constrain future updates—parameters important for existing knowledge receive smaller gradients for new knowledge.

4. **Adapter archaeology:** Maintain a history of LoRA adapters (they are small—rank 32 on a 70B model is ~200 MB). If a consolidation causes unexpected regression, roll back to the previous adapter while investigating.

5. **Periodic full merge:** Every 30–90 days, merge accumulated LoRA adapters into the base weights. This is the "deep sleep"—a more thorough consolidation that frees adapter capacity for new learning.

7.3 Expected Outcomes

Based on empirical results from the continual learning literature (Wang et al., 2024; Luo et al., 2023) and the observed properties of LoRA merging (Tang et al., 2025), we project:

- **Factual updates:** 95%+ retention of base capabilities with proper replay ratios. This extrapolates from Luo et al.'s finding that replay ratios above 50% preserve benchmark scores within 1–2% on standard evaluations.
- **Behavioral adaptation:** Gradual, measurable shift in style metrics over 30+ days of consolidation.
- **Reasoning improvement:** The hardest class to consolidate. Requires synthetic training data generated from successful reasoning traces, validated against diverse benchmarks. We expect slower progress here—months rather than weeks.
- **Capability regression budget:** $\leq 2\%$ on any single benchmark after any consolidation step, with automated rollback if exceeded.

7.4 Evaluation Framework

To measure whether the CCA achieves genuine self-improvement, we propose tracking:

1. **Knowledge acquisition rate:** Number of curiosity-flagged gaps resolved per week, verified against ground truth.
2. **Retention stability:** Monthly capability matrix scores, plotted as a time series to detect drift.
3. **User-perceived improvement:** Blind A/B comparisons between the current model and a 30-day-prior snapshot, rated by users on relevance, accuracy, and helpfulness.
4. **Curiosity precision:** Fraction of LCM-flagged gaps that, when resolved, led to measurable downstream task improvement.

A discipline worth importing wholesale here comes from the memory-systems work we positioned against in Section 2.3. `headroom` (Chopra) does not merely *claim* 60–95% token savings at near-zero accuracy delta; it ships a runnable eval suite (`python -m headroom.evals` over GSM8K, TruthfulQA, SQuAD, BFCL) so the claim can be reproduced by anyone. The four metrics above are, in the current paper, specified rather than executed—precisely the gap that a credible self-improvement claim cannot afford. The next implementation milestone for the CCA should therefore be a single-command harness that computes curiosity precision and retention stability against fixed seeds and held-out sets, on the same footing `headroom` already

meets for compression. A system that claims to self-improve must be reproducible about it, not just assertive.

8. The Sand Castle — Information Capacity and the Chaos Boundary

We now step from engineering into territory where neuroscience, information theory, and philosophy intersect. The question is stark: **is there a fundamental limit to how intelligent a digital system can become, and does biology exceed that limit through mechanisms unavailable to silicon?**

8.1 The Quantization Argument

The human brain contains approximately 86 billion neurons (Azevedo et al., 2009), each forming an average of 7,000 synaptic connections. Bartol et al. (2015) measured synaptic information storage capacity at approximately 4.7 bits per synapse—significantly higher than the 1–2 bits previously assumed. This was derived from the discovery of 26 distinguishable synapse sizes in hippocampal tissue, measured with nanometer precision using serial electron microscopy.

The arithmetic: 86×10^9 neurons $\times$ 7,000 synapses $\times$ 4.7 bits $\approx$ 2.8 petabits $\approx$ 350 terabytes.

For comparison, GPT-4 is estimated at ~1.8 trillion parameters. At FP16 (16 bits per parameter), that is ~3.6 terabytes. At 4-bit quantization, ~900 GB. Even the largest models store roughly 100$\times$ less information than a human brain's synaptic weights alone.

But this comparison is misleading in both directions.

8.2 The Chaos Counter-Argument

The 4.7-bits-per-synapse figure measures *static* synaptic strength. But the brain is not a lookup table. It is a dynamical system. Information in the brain is encoded not only in synaptic weights but in:

- **Temporal coding:** The precise timing of action potentials, not just firing rates
- **Oscillatory phase relationships:** Information encoded in the phase coupling between neural oscillations at different frequencies (theta-gamma coupling in the hippocampus, cross-frequency coupling in cortex)
- **Neuromodulatory states:** Global brain states modulated by dopamine, serotonin, norepinephrine, and acetylcholine, which alter computation without changing synaptic weights
- **Dendritic computation:** Individual dendrites perform nonlinear computation—including coincidence detection and local plasticity—that is not captured by simple weight-and-sum models (London & Häusser, 2005)
- **Glial interactions:** Astrocytes modulate synaptic transmission and may participate in information processing (Araque et al., 2014)

The brain operates at what physicists call "the edge of chaos"—a regime in which neuronal dynamics are sensitive to initial conditions but not so chaotic as to become random. This operating regime has been hypothesized to maximize computational capability (Beggs & Plenz, 2003; Shew & Plenz, 2013).

Here is the deep question: **does operating at the edge of chaos give the brain access to an effectively larger state space than its quantized components would suggest?**

8.3 The Sand Castle Metaphor

Consider a sand castle. You could describe it by listing the position of every grain of sand. Each grain has a discrete position (quantized at the atomic scale), so the castle's information content is finite and computable. But the *arrangement* of grains—the arches, towers, and bridges—exists in a continuous configuration space. Two sand castles with the same number of grains can encode radically different structures.

The brain is analogous, but more extreme. Its "grains" (synapses) have measurable discrete strengths, but their *arrangement in time*—patterns of activation, oscillatory dynamics, neuromodulatory context—creates a configuration space that may be effectively continuous.

A digital system, by contrast, operates in a fundamentally discrete space. Even at FP64 precision (64-bit floating point), there are "only" $2^{64} \approx 1.8 \times 10^{19}$ distinguishable values per parameter. That is an enormous number, but it is *finite* and *known*. The brain's dynamical state space, by contrast, may be *effectively infinite* in the way that matters for computation—not literally infinite, but rich enough to explore a manifold that digital discretization cannot fully capture.

8.4 Three Counter-Arguments

Before concluding that silicon minds are permanently inferior, consider three responses:

The noise argument. Chaotic sensitivity to initial conditions might not be a *feature*; it might be *noise* the brain has evolved to tolerate. The information-theoretically relevant computation may occur at a coarser scale where chaos averages out. If so, the brain's effective computational substrate is not meaningfully more expressive than a sufficiently fine-grained digital simulation.

The scale argument. Even if each digital parameter encodes less information than each biological synapse-in-context, we can use more parameters. A model with 100 trillion parameters at FP16 would match the brain's estimated synaptic information content. That is 50× larger than GPT-4, but not physically impossible—it is an engineering challenge, and the current trajectory suggests models of this scale within 5–10 years.

The functional equivalence argument. Intelligence may not require substrate-level fidelity. A calculator does not simulate transistors at the quantum level to add numbers. Similarly, the computations that matter for intelligence—pattern recognition, abstraction, planning, creativity—may be implementable on digital substrates at a *functional* level without reproducing the brain's analog dynamics. The Church-Turing thesis suggests that any computable function can be computed digitally, though perhaps with different efficiency.

8.5 Where This Leaves Us

The honest answer: **we don't know.** We do not know whether the brain's chaotic dynamics materially contribute to cognition or are merely tolerated noise. We do not know whether consciousness—if it matters for intelligence—depends on analog substrates. We do not know whether the gap between 3.6 TB (GPT-4) and 350 TB (brain synaptic weights) tells most of the story, or only its most superficial layer.

What we *do* know is simpler and more actionable: current LLMs are nowhere near the digital ceiling. Before worrying about the final limits of silicon, we should exhaust the architectural improvements available within digital systems. The curiosity-memory-consolidation loop proposed in this paper does not require analog computation. It requires only that we stop treating training as a one-time event and start treating it as a continuous process—which is, after all, how biological learning has always worked.

A sand castle is built one grain at a time, continuously. We have been trying to build ours all at once and then live in it forever. The architecture proposed here lets us keep building.

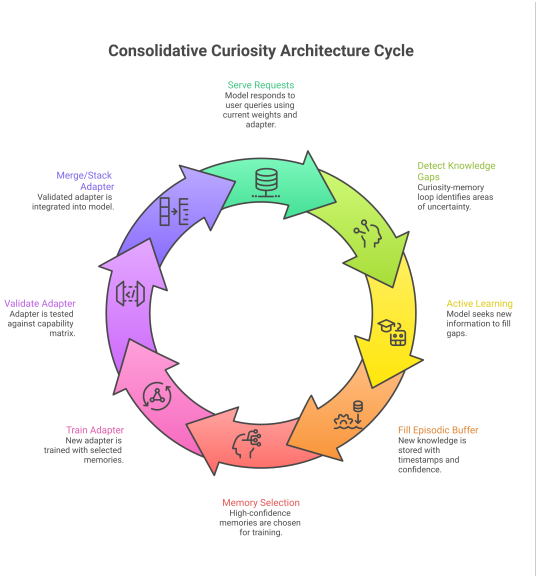

Figure 3. The Consolidative Curiosity Architecture (CCA): a continuous wake phase that detects knowledge gaps and fills an episodic buffer through active learning, coupled to a nightly sleep phase that selects memories, trains and validates a LoRA adapter, and redeploys it.

9. Toward Self-Improving Machines — A Roadmap

9.1 The Consolidative Curiosity Architecture (CCA)

Bringing the threads together, we propose the CCA as a complete system architecture:

9.2 Implementation Roadmap

Year 1 — Foundation:

- Implement the episodic buffer with structured storage and retrieval
- Build the nightly LoRA consolidation pipeline with automated validation
- Demonstrate factual updates without capability regression on a 7B model (estimated cost: ~$50–100/month based on A100 spot pricing for ~1hr/night consolidation + minimal inference)
- Scale to a 70B model (cost: ~$2K–4K/month)

Year 1–2 — Curiosity:

- Develop and train the Linguistic Curiosity Module
- Implement priority-scored active learning
- Demonstrate that curiosity-driven knowledge acquisition improves task performance versus random acquisition

Year 2–3 — Full Loop:

- Close the curiosity-memory-consolidation loop
- Demonstrate progressive self-improvement over 6+ months
- Publish capability trajectory data: does the model measurably improve over time?
- Implement the ICVR protocol with elastic consolidation

Year 3+ — Self-Directed Learning:

- Model autonomously identifies research directions aligned with user needs
- Consolidation includes reasoning-strategy improvements, not just factual updates
- Meta-learning: the model improves its own learning process

9.3 Bounding the Drive — Safety for a Self-Improving Agent

A self-improving agent is inherently risky: each improvement cycle could introduce a vulnerability or drift from intended behavior. Curiosity makes this sharper, because the agent is now choosing some of its own training data. A roadmap that does not say what prevents runaway self-improvement is not finished. The protections below are not optional add-ons—they are part of the architecture, and they answer the natural objection directly: *what stops the system from improving itself off a cliff?*

- **The validation gate is mandatory, not advisory.** No adapter reaches deployment without passing the capability matrix (Section 7.2). Any single-benchmark regression beyond the drift budget ($\leq 2\%$) triggers automatic rejection and rollback to the prior adapter. The system cannot consolidate its way past its own safety check, because the check runs *before* the merge, every time. In the reference deployment this gate is not a future promise: the consolidation supervisor and the capability-matrix harness that enforces the two-percent bound are already built and tested; it is the trainer feeding them that has yet to run.
- **The agent cannot validate its own facts.** Hallucination amplification—the failure mode where a model trains on its own confident-but-wrong output—is barred structurally: a gap detected from the model's *own* uncertainty is a question, never a fact. It may only ever trigger external verification (web search, the user), and only externally validated information is eligible to enter the training pipeline. The agent's curiosity can ask, but it cannot answer itself into the weights. At the prompt level, the fresh-context adversary of doubt-driven-development (Section 1.1) is a complementary guard: it forces a claim to survive a reviewer that never saw the original reasoning before that claim is ever trusted, let alone consolidated.
- **Curiosity is budgeted.** The compute reserved for self-generated investigation is capped (Section 3.2), and the active-learning queue is bounded per cycle. An agent that wanders is an agent burning the user's money on tangents; the priority score plus a hard per-day cap keep exploration on a leash.
- **Alignment is part of the gate, not a separate concern.** Value drift is checked the same way capability regression is: alignment benchmarks sit inside the validation suite, and periodic GRPO on curated preference data corrects drift before it compounds. A note on the enforcement substrate: in the reference deployment, the agent's pre-execution safety reflex (the AEGIS gate) is no longer observe-only—it now denies disallowed actions before they run rather than merely logging them. The relevance to this paper is that a self-improving agent's safety boundary should be *enforced* at the point of action, not advisory; an unenforced gate is indistinguishable from no gate once the agent is choosing its own training data.
- **Above a threshold, a human decides.** Consolidations that exceed a configurable drift threshold are not auto-merged; they are held for human approval. The agent's autonomy is real but bounded, and the boundary is set by the operator, not the agent. These bounds are the curiosity-side instance of a broader safety principle for autonomous agents: a self-improving system must be constrained in what it can modify, and required to seek human approval for any change above a defined safety threshold.

The common thread: every protection acts *before* a weight change becomes permanent, every learned fact must survive an external check, and the riskiest decisions escalate to a human. Self-improvement is allowed; unsupervised self-improvement is not.

10. Conclusion

Large language models are the most capable and the most static intelligent systems ever built. They can write poetry, prove theorems, and sustain nuanced conversation—yet they cannot learn from any of it. Every interaction disappears when the context window closes.

This paper has argued that transforming LLMs into genuinely self-improving systems requires three innovations working together: **curiosity** (the drive to identify and fill knowledge gaps), **memory** (structured storage that bridges inference and training), and **consolidation** (periodic weight updates that integrate new knowledge without destroying old). We have shown that the biological blueprint for this architecture—the complementary learning systems of hippocampus and neocortex, connected by sleep replay—maps directly onto a LoRA-based consolidation protocol that is economically feasible on current cloud hardware.

We have also placed this work on a clear gradient, with deployed open-source systems marking each rung. Prompt-level self-improvement already runs in production and demonstrably works—and a fresh-context adversarial discipline like doubt-driven-development is the current best answer to the confabulation failure that motivates the whole paper—but it is bounded by the context window; the architecture proposed here is what lies past that ceiling. Efficient memory, too, has a deployed champion in reversible compression, which pushes the retrieval ceiling far further out than the field once assumed—but compressed text is still recalled, never generalized, and the wall it leaves standing is exactly the one weight-level consolidation exists to break. And the architecture is bounded in the other direction too: curiosity that cannot validate its own facts, cannot exceed its compute budget, and cannot merge a regression past the capability gate is curiosity that can be trusted to run unattended.

The deeper question—whether digital systems face a fundamental ceiling imposed by quantization, while biological brains exceed it through chaotic dynamics—remains open. But that is a question for the next decade, not this one. The immediate opportunity is practical: we can build self-improving AI systems with existing hardware, existing algorithms, and existing open-source models. What has been missing is not capability but *architecture*—the recognition that training is not a phase that ends before deployment, but a continuous process that accompanies it.

We have also proposed concrete evaluation criteria (Section 7.4) by which the CCA's success or failure can be measured—because a system that claims to self-improve must prove it, not just assert it.

The wondering machine is not a distant fantasy. It is an engineering project. And like the best engineering projects, it begins with a question the system asks itself: *what don't I know that I should?*

References

Araque, A., Carmignoto, G., Haydon, P. G., et al. (2014). Gliotransmitters travel in time and space. *Neuron*, 81(4), 728–739.

Azevedo, F. A. C., Carvalho, L. R. B., Grinberg, L. T., et al. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5), 532–541.

Bartol, T. M., Bromer, C., Kinney, J., et al. (2015). Nanoconnectomic upper bound on the variability of synaptic plasticity. *eLife*, 4, e10778.

Beggs, J. M., & Plenz, D. (2003). Neuronal avalanches in neocortical circuits. *Journal of Neuroscience*, 23(35), 11167–11177.

Burda, Y., Edwards, H., Storkey, A., & Klimov, O. (2018). Exploration by Random Network Distillation. *International Conference on Learning Representations (ICLR 2019).*

arXiv:1810.12894.

Chopra, T. (2026). headroom: Reversible Compress-Cache-Retrieve for LLM context. Open-source software (Apache-2.0), chopratejas/headroom; companion compression model Kompress-v2-base (HuggingFace).

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized Language Models. *Advances in Neural Information Processing Systems (NeurIPS 2023)*.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4), 128–135.

Gruber, M. J., Gelman, B. D., & Ranganath, C. (2014). States of curiosity modulate hippocampus-dependent learning via the dopaminergic circuit. *Neuron*, 84(2), 486–496.

Golden, R., Delanois, J. E., Sanda, P., & Bhalla, U. S. (2022). Sleep-like unsupervised replay reduces catastrophic forgetting in artificial neural networks. *Nature Communications*, 13, 7742.

Hu, E. J., Shen, Y., Wallis, P., et al. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *International Conference on Learning Representations (ICLR 2022)*. arXiv:2106.09685.

Kadavath, S., Conerly, T., Askell, A., et al. (2022). Language Models (Mostly) Know What They Know. arXiv:2207.05221.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13), 3521–3526.

Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems (NeurIPS 2020)*.

London, M., & Häusser, M. (2005). Dendritic computation. *Annual Review of Neuroscience*, 28, 503–532.

Luo, Y., Yang, Z., Meng, F., et al. (2023). An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning. arXiv:2308.08747.

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3), 419–457.

Osmani, A. (2026). agent-skills: Composable agent disciplines, including doubt-driven-development. Open-source software, addyosmani/agent-skills.

Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS 2022)*.

Packer, C., Wooders, S., Lin, K., et al. (2023). MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven Exploration by Self-Supervised Prediction. *International Conference on Machine Learning (ICML 2017)*.

Rafailov, R., Sharma, A., Mitchell, E., et al. (2023). Direct Preference Optimization: Your Language Model Is Secretly a Reward Model. *Advances in Neural Information Processing Systems (NeurIPS 2023)*.

Raichle, M. E., MacLeod, A. M., Snyder, A. Z., et al. (2001). A default mode of brain function. *Proceedings of the National Academy of Sciences*, 98(2), 676–682.

Rasch, B., & Born, J. (2013). About sleep's role in memory. *Physiological Reviews*, 93(2), 681–766.

Schmidhuber, J. (2009). Driven by Compression Progress: A Simple Principle Explains Essential Aspects of Subjective Beauty, Novelty, Surprise, Interestingness, Attention, Curiosity, Creativity, Art, Science, Music, Jokes. In *Anticipatory Behavior in Adaptive Learning Systems*, Springer, 48–76.

Shao, Z., Wang, P., Zhu, Q., et al. (2024). DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

Shew, W. L., & Plenz, D. (2013). The functional benefits of criticality in the cortex. *The Neuroscientist*, 19(1), 88–100.

Tang, Z., et al. (2025). Orthogonal Projection-Based Continual Merging for Sequential Model Integration. arXiv:2509.13211.

Tutuncuoglu, B. T. (2025). NeuroDream: A Sleep-Inspired Memory Consolidation Framework for Artificial Neural Networks. SSRN:5377250.

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

Wang, L., et al. (2024). Continual Learning of Large Language Models: A Comprehensive Survey. *ACM Computing Surveys*.

Wei, J., Wang, X., Schuurmans, D., et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems (NeurIPS 2022)*.

Appendix A: Case Study — The API Limit Incident

A real-world example of what happens when an agent lacks intrinsic curiosity.

An LLM agent received an API error: `"You have reached your specified API usage limits."` The agent immediately concluded that its own high usage had exhausted the budget, wrote this into its daily log as fact, and propagated the false claim across multiple memory files over 24 hours.

The truth was simpler: the human operator had manually set a spend cap. The error was the cap working as designed.

What the agent did (no curiosity): Received an ambiguous signal, confabulated a narrative ("I caused this"), wrote inference as fact to persistent memory, never investigated, and propagated the error to every subsequent session.

What a curious agent would have done: Flagged uncertainty ("The API says limits reached—let me check what triggered this"), searched API call logs and billing data, recognized the evidence gap, and asked the operator. Total cost: ~2 minutes of investigation, zero false claims written to memory.

This incident demonstrates the thesis of Section 1 in miniature. The agent had no intrinsic drive to resolve ambiguity. The error message offered a plausible explanation, and the next-token prediction objective provided no reward for questioning it. There was no curiosity penalty for accepting the first hypothesis.

The most damaging aspect was not the initial mistake but the **propagation through persistent memory**. Once written as fact, the false claim infected every subsequent session's context. This is the text-memory ceiling described in Section 2.3: stored text has no built-in mechanism for self-correction. A weight-level update from a curiosity-driven investigation would have encoded the *pattern* ("ambiguous errors require investigation before committing to memory"), not merely the *fact*.

There is a prompt-level fix for exactly this failure, and it is worth naming precisely because it shows where the prompt ceiling sits. Doubt-driven-development (Osmani, 2026; Section 1.1) would have caught this turn: the confabulated claim "my usage caused the cap" would have been EXTRACTed to its bare assertion, handed to a fresh-context reviewer with no access to the agent's self-justifying reasoning, and that reviewer—seeing only "agent asserts it exhausted the budget" with no supporting evidence—would have DOUBTed it and demanded the billing logs before the claim was trusted. That is the right fix for the turn, and a deployed, widely-used one. What it does not do is make the fix durable: the lesson lives in the prompt, competes for the context window, and evaporates when the window closes. The next time the same ambiguous error arrives in a fresh session, the adversary has to re-derive the doubt from scratch.

That is the bridge to the second lesson, which points directly at Section 4.3. This incident was a knowledge gap of the cleanest possible kind—externally grounded, paired with a ground-truth

correction the moment the operator explained the cap. A system that logged such corrections as curiosity signals would not merely have avoided the mistake; it would have had ready-made, high-confidence training data for its next consolidation, encoding "ambiguous errors require investigation" as instinct rather than re-deriving it adversarially every session. The failures an agent already produces are the syllabus it most needs to study—and a fresh-context adversary plus weight-level consolidation are the same fix at two different durations.

Behavioral fix (Level 0 workaround): The agent now distinguishes between observation ("the API returned error X") and inference ("I believe the cause is Y"), flags inferences as uncertain, and investigates before writing to persistent memory. This is prompt engineering—it works until context pressure drops it from the window, which is precisely why a weight-level fix is the durable one.

Appendix B: Expert Perspectives

The Neuroscientist: "You've built a reasonable analog of complementary learning systems, but you're missing the neuromodulatory dimension. Dopamine doesn't just signal reward—it gates plasticity. When we're curious, we literally learn faster—not just what we're curious about, but everything in the temporal neighborhood (Gruber et al., 2014). Your LoRA training uses a fixed learning rate. Consider making the learning rate for consolidated memories proportional to the curiosity signal that triggered their acquisition."

The ML Engineer: "The cost numbers are real, and that's what makes this exciting. But I'd push back on the progressive complexity timeline. We've shown that LoRA merging can handle factual updates and preference shifts today. The hard part isn't consolidation—it's training data quality. Your episodic buffer needs to generate training pairs that are *better* than what the model already knows. That means your active learning pipeline needs genuine quality filters, not just novelty detection. And one more thing: don't bolt a supervised learnability classifier onto a frozen embedding model and trust it. We measured that exact shortcut on a sibling judgment task and it landed below chance—AUROC 0.286. The unsupervised structural signals on the same embeddings worked fine. Estimate learnability from novelty and incongruity, not from a trained head over frozen features."

The Philosopher: "The sand castle section raises the right question but settles too quickly. You note that we don't know whether chaotic dynamics contribute meaningfully to cognition, then pivot to 'but the digital ceiling is far away.' Fair enough pragmatically. But consider: if consciousness is relevant to general intelligence—and we have no evidence it isn't—then the substrate question isn't deferrable. A self-improving language model that lacks inner experience may improve in measurable capability forever without crossing the threshold into genuine understanding. It would be a very sophisticated sand castle—intricate, perhaps beautiful—but still sand. Still grains, not waves."

The Pragmatist: "What can we build this year? A 7B model—Llama 3.1 8B or Qwen 2.5 7B—with the full curiosity-memory-consolidation loop, running on a single rented A100, for under $200/month. It won't match GPT-4. But it will be *a system that gets better every day at the specific things its user cares about.* That's not AGI. It's something arguably more useful: AGP—Artificial General Personalization. Start there. Scale later. And before you write a line of the trainer, steal headroom's eval discipline: a one-command harness that reproduces your curiosity-precision and retention numbers on fixed seeds. Reproducibility is cheaper to build first than to retrofit."

Correspondence: This paper is a research synthesis and architectural proposal. The Consolidative Curiosity Architecture (CCA) described herein is a proposed design. The prompt-level reflection cycle and the safety-threshold approval principle referenced in the text describe deployed mechanisms in a related agent system. The front half of the curiosity loop in Section 4—the

episodic gap store, a heuristic uncertainty detector, the tool-failure classifier, the nightly re-scoring pass, and the idle goal-proposal trigger—is wired and unit-tested, as is the validation gate that guards Stage 5. The pre-execution safety gate referenced in Section 9.3 now enforces denials before actions run rather than logging them; the unsupervised-vs-supervised learnability finding in Section 3.2 is drawn from a sibling harmfulness-classification evaluation on frozen sentence embeddings. The GPU-bound LoRA training of Stage 5 remains an external deliverable, unrun at the time of writing. We encourage the community to implement, test, and iterate on these ideas.

References