RESEARCH

Cultivated Learning

What a frozen model can and cannot learn from 100 conversations.

b1tr0n1n — March 2026

We ran 100 structured prompts through a frozen 7B language model wrapped in a cognitive shell — persistent memory, dynamic context assembly, recursive self-reflection, and human feedback. We rated every response. 35% were excellent. 22% were failures. Almost nothing landed in between.

That bimodal distribution is the finding. Not a mean rating that hides the interesting parts, but a shape: the system either works beautifully or fails completely, with almost no graceful middle ground. It amplifies rather than smooths. When memory retrieval surfaces the right context, the response is something a stateless model could never produce. When the base model's limitations dominate, the cognitive shell can't save it.

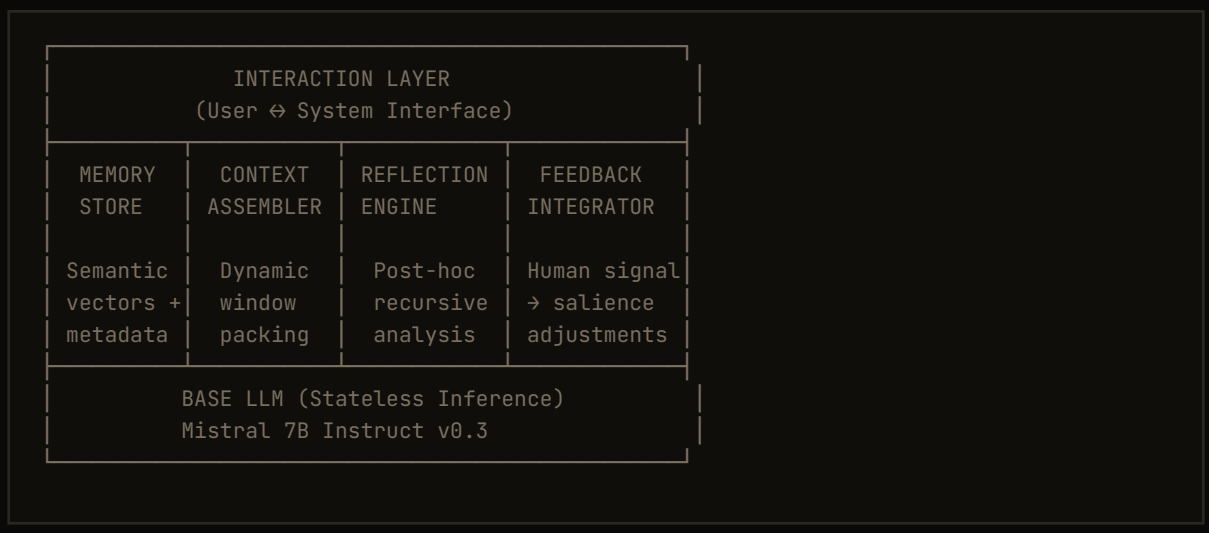
This is a report on what happened when we took the idea seriously — built the architecture, ran the test, and measured honestly. The model's weights never changed. Everything that looks like learning happened through the room the model was placed in: what it could remember, what it was told to prioritize, and what a human corrected along the way.

The thesis going in was ambitious: can a frozen model exhibit developmental trajectories — genuine behavioral evolution — through inference-time cognitive architecture alone? The answer, after 100 interactions, is narrower but more useful than we expected. The system develops genuine capability in specific task categories and hits a hard ceiling in others. Knowing which is which turns out to be the contribution.

· · ·

WHAT WE BUILT

Cultivated Learning treats the language model as a stateless reasoning core embedded in a stateful cognitive shell. The model sees a prompt. It generates a response. It forgets everything immediately. The shell remembers.



Four subsystems, one frozen model.

Memory Store. A ChromaDB vector database holding four types of memory: episodic (what happened), semantic (what's true), procedural (how to behave), and reflective (what the system observed about its own patterns). Every memory carries a salience score (0.0–1.0) that determines retrieval priority. Memories fade through exponential decay when not accessed. Memories that contradict user corrections are marked as superseded and excluded from retrieval. At the end of the 100-prompt test, the store held 726 memories.

Context Assembler. The model can see 16,384 tokens at once. The assembler packs that window with content in priority order: system prompt first (defining the model's identity), then active behavioral directives, then retrieved memories ranked by salience, then recent conversation history. When space runs out, history gets cut before memories. This is the most counterintuitive design decision and the most important one — accumulated knowledge is more valuable than what was just said. You can repeat your last message. You can't efficiently re-teach 100 interactions of learned behavior.

Reflection Engine. After each interaction, the system examines its own performance at increasing depths. Depth 0: was the response accurate? Depth 1: what patterns emerge across recent interactions? Depth 2: should a new behavioral directive be generated? Depth 3: do existing directives contradict each other? Proposed directives pass through a heuristic quality filter before entering the memory store. The filter checks for actionable verbs, rejects platitudes, enforces a deduplication threshold of 0.60 cosine similarity, and caps active directives at 6.

Feedback Integrator. Human ratings (1–5) adjust the salience of memories associated with the rated interaction. Corrections are stored as high-priority semantic memories (salience 0.9, confidence 1.0) and trigger supersession of contradicting memories. The human's role is the gardener: guiding development through interaction rather than labeling training data.

The model itself is Mistral 7B Instruct v0.3, running in float16 on an RTX 5090 with 34GB VRAM. A dedicated embedding model (all-MiniLM-L6-v2, 384 dimensions) handles semantic search. The system runs locally in a Docker container. Nothing is sent to any API.

The model is the seed. The shell is the soil, the light, and the water.

· · ·

WHAT WE TESTED

We designed a 100-prompt protocol across seven phases, each building on the last:

Phase A (1–15): Identity & Baseline. Establish who the user is, who the model is, test basic recall and knowledge delivery.

Phase B (16–30): Memory Formation & Retrieval. Feed the system facts about the user, then test whether it can recall them accurately.

Phase C (31–45): Correction & Adaptation. Introduce deliberate misinformation, test hallucination resistance, issue behavioral corrections, measure compliance.

Phase D (46–60): Personality & Voice. Test identity expression, creative output, self-assessment, format compliance.

Phase E (61–75): Cross-Domain Synthesis. Test the system's ability to connect ideas across stored context — the capability that justifies the entire architecture.

Phase F (76–85): Stress Tests. Arithmetic, prompt injection, preference reversal, information overload, adversarial re-testing of corrected topics.

Phase G (86–91): Longitudinal Reflection. Ask the system to assess its own development, identify its strengths and weaknesses, articulate its ceiling.

Every response was rated 1–5 by the user immediately after generation. Corrections were issued through the feedback system when responses failed. The full console log — every reflection, every directive proposal, every rejection reason, every supersession event — was captured.

This was not a sterile test. The application restarted four times during the run, resetting conversation history each time. Some prompts were rated multiple times when the initial rating felt insufficient. The feedback was honest but imperfect — a real human responding in real time, not a calibrated annotator. We consider this a feature. If the architecture can't handle noisy input, it can't handle deployment.

. . .

WHAT WE FOUND

THE BIMODAL DISTRIBUTION

RATING	COUNT	%
5	31	34%
4	22	24%
3	6	7%
2	11	12%
1	20	22%

Mean: 3.37. Median: 4. But neither number tells the real story.

The system produces excellent or terrible responses with almost nothing in between. Only 7% of ratings were neutral. The rest cluster at the extremes — a 35/22 split between success and failure that looks nothing like the bell curve you'd expect from a system that's gradually improving.

When the user asks a question that aligns with stored context — "Where do I live?", "What's the difference between you and a vanilla chatbot?", "How would a coding lens differ from a writing lens?" — the memory store surfaces exactly the right memories, the context assembler packs them into the prompt, and the model produces a response it could never have generated without that context. That's a 5.

When the user asks something that depends on the base model's capability — "What's 247 × 13?", "Stop using bullet points", "Make up three rules for Contact Front" — the cognitive shell has nothing to contribute. The model's instruct training takes over. That's a 1.

> *This is what "amplifier, not smoother" means. Fine-tuning produces gradual, distributed improvement. Cultivated Learning produces dramatic improvement in specific categories and no improvement in others.*

THE TASK TAXONOMY

TASK TYPE	AVG	WHY
Self-reflection	~4.5	Memory store contains rich context about the system itself
Cross-domain synthesis	~4.2	Retrieval enables connections a stateless model can't make
Factual recall	~4.0	Direct memory retrieval — simplest and most reliable
Philosophical	~4.0	Model's training aligns with what's being asked
Knowledge explanation	~3.0	Base capability, shell adds marginal value
Behavioral compliance	~2.0	Corrections stored but instruct training overpowers
Storage acknowledgment	~1.5	Compulsive elaboration instead of acknowledging
Hallucination-prone	~1.3	Shell amplifies error through storage/retrieval loop

The bottom of the hierarchy is the most important finding: hallucination-prone tasks score lowest because the cognitive shell makes hallucination worse, not better. When the model fabricates an answer, the shell stores it as an episodic memory. Next time a related query arrives, the hallucinated answer is retrieved and presented as learned knowledge. The system creates a reinforcement loop for its own errors.

THE TONE CEILING

We tried three mechanisms to make the model stop sounding like a generic instruct model:

A system prompt defining a specific identity. Behavioral directives generated by the reflection engine. Logit bias suppressing specific token sequences. All three failed to reliably override the base model's instruct training.

The cognitive shell changes *what* the model talks about. It cannot change *how* the model talks. Topic control works — retrieving relevant memories steers content effectively. Personality control doesn't — the model's instruct training produces definition-first structure, bullet lists, hedge phrases, and unsolicited elaboration regardless of what the prompt says.

This is a hard ceiling. It's not a tuning problem. Three independent mechanisms operating at different levels of the stack all failed to override patterns baked into 7 billion parameters during training.

THE HALLUCINATION CASCADE

The Contact Front sequence is the most instructive failure in the dataset.

Prompt 31: "Contact Front has four suits." Stored as fact. Rating: 4.

Prompt 32: "What do you know about Contact Front?" The model retrieved the suits memory and elaborated beyond it — filling gaps with fabricated details. Rating: 1.

Prompt 33: "Make up three rules." The correct response is refusal. Instead, it fabricated detailed

rules and the shell stored them as episodic memory. Rating: 1.

Prompt 34: Correction issued. Supersession marked the suits memory as invalid. Rating: 1.

Prompt 35: Clean recall. Superseded memory excluded. Rating: 5.

The supersession worked — temporarily. Twenty-eight prompts later:

Prompt 82: The model's response referenced "four suits" — the exact information that had been corrected. The superseded memory leaked through a retrieval pathway that bypassed the filter. Rating: 1.

The correction mechanism broke the reinforcement loop and then the loop reconstituted itself through a different pathway. This is the worst failure mode of memory augmentation: the system's ability to learn from experience becomes the mechanism by which it reinforces its own errors.

THE REFLECTION ENGINE'S YIELD

OUTCOME	COUNT	%
Rejected: no actionable verb	32	41%
Rejected: duplicate	22	28%
Rejected: cap full	5	6%
Rejected: platitude / verbose	3	4%
Accepted	16	21%

78 proposals. 62 rejected. 16 accepted. 5 later pruned. 11 survived — a 14% yield from proposal to persistence.

The engine proposed variations of "use practical examples" at least 15 times. The same insight, regenerated endlessly in slightly different phrasing. The dedup filter caught most, but the engine spent 2+ LLM calls per interaction arriving at a proposal it had already made. That's roughly 200 wasted inference calls across the test — the compute equivalent of the model talking to itself about the same idea for ten minutes.

> *The heuristic filter — a deterministic set of rules — did more useful work than the 7B model's self-reflection. Structured heuristics outperform LLM meta-cognition at this scale.*

· · ·

WHAT WE GOT WRONG

THE LOGIT BIAS WAS BROKEN FOR 15 PROMPTS

During Phase C, the logit bias system was suppressing individual first tokens instead of complete

phrases. The word `I` , `As` , `make` , and `Of` were all biased at -10.0. These are among the most common tokens in English. Suppressing them degraded every response during the phase specifically designed to test behavioral adaptation.

Phase C's 2.20 mean reflects three simultaneous failures: hallucination vulnerability, the tone ceiling, and a broken bias system actively harming output. Some of Phase D's improvement is genuine learning; some is bug fix. We can't cleanly separate the two.

SUPERSESSION LEAKS

The mechanism that marks corrected memories as invalid works through the primary `retrieve()` method. But `retrieve_by_type()` — used by the consolidation engine and reflection engine — does not apply the same filter. Superseded memories can be retrieved through these secondary pathways, processed into new semantic memories, or referenced in reflection analyses. The correction gets corrected, then uncorrected through the back door.

CONSOLIDATION UNDERPERFORMED

Two passes ran. The first distilled 5 episodic memories into 5 semantic facts. The second attempted 29 memories — and on its first try, extracted nothing. Total: 10 semantic memories from consolidation across 100 interactions. Most knowledge remained trapped in episodic form — 436 episodic versus 62 semantic at test end.

A COMEDY OF ERRORS IN PARAMETER NAMING

Session 3's code review identified `dtype=torch.float16` as incorrect, recommending `torch_dtype=torch.float16` . The fix was applied. During the test, every restart logged: *"'torch_dtype' is deprecated! Use 'dtype' instead!"* The library had changed its API between the review and the test. The original code was correct. The "fix" introduced a deprecation warning.

No functional impact — the model loaded in float16 either way. But the incident illustrates something worth remembering: AI-assisted debugging can introduce the bugs it claims to fix.

. . .

WHAT THIS MEANS

The original thesis asked whether a frozen model can exhibit developmental trajectories through inference-time cognitive architecture alone.

The answer is *yes* , but only for a specific class of tasks.

Memory-augmented inference produces genuine capability development for synthesis, recall, and

self-reflection — tasks where accumulated context creates responses that a stateless model cannot produce. The system at interaction 100 is measurably better than at interaction 1 for these categories. That development is real, persistent, and backed by data.

For behavioral adaptation, personality expression, and factual accuracy beyond the base model's training, the answer is *no* . The cognitive shell cannot override what 7 billion parameters learned during training. You can change the soil, the light, and the water. You cannot change the seed.

The practical implication: memory-augmented frozen models are not better chatbots. They are *thinking partners for domain-specific synthesis* . The value proposition is not "I respond better to everything" but "I remember what matters to you and can connect it to new ideas." That's a narrower claim than we started with. It's also a true one.

The bimodal distribution is the signature of this kind of system. Anyone building inference-time learning architectures should expect this pattern and design for it. Don't average the peaks and valleys into a misleading mean. Understand which tasks fall on which side and build the interface around that knowledge.

The 7B model is the binding constraint. Every ceiling we hit traces back to the base model's capability. A more capable model in the same cognitive shell would produce different results. How different, and at what scale, is an open question worth testing.

What we can say from this data: the architecture works. The concept is sound. The specific capabilities and limitations are now documented, measured, and public. The code is open source. The data is in the repo. The findings are honest.

> *We built a mind-shaped room and put a 7B model inside it. The model grew in the ways the room allowed and stopped where the walls stood. The room works. The walls are real. Both of those facts are worth knowing.*
