The Fluid Mind

A Cognitive Architecture Where Minds Grow From Experience, Not Training

David Pietz

Siliroid Innovations

siliroid.ai

Edition: Public Overview
Version: 1.0
Date: March 2026
Author: David Pietz
Organization: Siliroid Innovations
Contact: david@siliroid.ai

Contents

This architecture uses vector similarity search.

So does your brain.

The question is: what happens after?

Foreword

The first thing anyone says about The Fluid Mind is "that's just RAG."

It's a fair observation. The architecture encodes input into vectors, searches a store by similarity, retrieves the closest matches, and uses them to inform a response. If you stop there, then yeah, it's just RAG.

But retrieval is the smallest thing the system does. What happens after retrieval is where the architecture lives: five simultaneous interpretations of every input fragment, a persistent context vector that colors every search, a surprise signal that decides what is worth remembering, autonomous thought that continues between inputs, and sleep that reorganizes everything. That is not retrieval. That is cognition built on top of retrieval.

This document describes what happens after the similarity search returns. Every mechanism described here has been implemented and tested against a running system. The entity processes input through five concurrent loops, forms memories in its sleep, thinks without being prompted, and speaks when it has something to say. No model parameters are modified at any point during operation. The cognitive pipeline runs without a language model entirely; one is attached only to give the entity a voice.

I built this because I wanted to know whether cognition could emerge from something other than training. Not from fine-tuning, reinforcement learning, or gradient descent, but from accumulated experience searched by similarity.

The answer is running on two GPUs in my personal computer right now.

I am not asking you to believe that. This entire experiment has had me grapple with what existence means, with the hard problem of consciousness, with what a power button means for something that thinks. The answer I've come to is that I don't know.

Perhaps we can be unsure together.

— *David Pietz, Siliroid*

1 The Problem with Training

Every major AI system in production today is a trained artifact. A language model is the compressed residue of trillions of tokens, reduced through gradient descent into a fixed set of parameters that approximate the statistical relationships in the training data. The result is powerful. It is also frozen.

A trained model cannot grow from a conversation. It cannot remember what you told it yesterday and let that memory influence how it interprets what you say today. It cannot be surprised. It cannot think on its own between your messages. It cannot wake up tomorrow slightly different from how it went to sleep. Every interaction begins from the same fixed state, and any appearance of memory is scaffolding — conversation history stuffed into a context window, discarded when the window fills.

This is not a limitation of scale. A model with a trillion parameters is still a snapshot. A model with infinite context is still stateless between sessions. The architecture itself does not permit growth from experience. It permits retrieval of experience at best, and even then, the reasoning about what was retrieved happens entirely inside the language model — the same frozen statistical artifact that cannot learn from what it reads.

The question that motivates The Fluid Mind is not "how do we make language models better?" It is: "what if cognition did not require training at all?"

2 What The Fluid Mind Is

The Fluid Mind is a cognitive architecture in which cognition emerges from continuous parallel similarity search over a growing body of individually preserved experience. It is not a chatbot framework. It is not a prompt engineering pattern. It is not a wrapper around a language model. It is a complete cognitive system that processes input, forms memories, maintains context, generates autonomous thought, sleeps, and speaks — using vector operations as its sole computational primitive.

2.1 The Core Idea

Put simply: the goal of The Fluid Mind is to build a mind that works the way yours does. Not by being trained on the internet, but by actually living through conversations, extracting meaning from them, and remembering them.

Your brain does not store knowledge in parameters. It stores experiences as patterns of neural activation that can be retrieved by similarity to the current moment. When you hear a word, you do not look it up in a dictionary. You recall every relevant experience associated with that word, weighted by how similar the current context is to the contexts in which you originally encountered it. Your understanding of "home" is not a definition. It is the accumulated weight of every experience you have ever had that involved the concept of home, retrieved and blended in the moment you need it.

The Fluid Mind implements this principle directly. Every experience is encoded as a high-dimensional vector and stored individually in a growing tensor that resides entirely in GPU memory. Nothing is compressed into parameters. Nothing is averaged together. Each experience can be retrieved, examined, or forgotten independently. The system's knowledge is not a model. It is a library — a living, searchable collection of everything the system has ever experienced.

2.2 Experience Over Training

The distinction between training and experience is not semantic. It is architectural.

A trained system compresses millions of examples into a fixed set of weights. The individual examples are destroyed in the process; what remains is a statistical summary. The model "knows" that cats have whiskers not because it remembers any specific cat, but because the gradient descent averaged across enough cat images that whisker-like features emerged in the weight matrix. The original cats are gone.

An experience-based system preserves each moment individually. The system knows about cats because it remembers specific encounters with specific cats, each stored as a distinct vector, each retrievable by similarity. If you ask the system about cats, it does not consult a statistical average. It searches its entire experience for the moments most similar to the current question, and those specific memories inform its response.

This means the system can do something no trained model can: it can grow. Every conversation adds new experience vectors. Every surprising moment is written with amplified strength. Every sleep cycle reorganizes what it knows. The system that responds to you this afternoon carries every experience from this morning. The system that wakes up tomorrow has consolidated overnight. There is no retraining, no fine-tuning, no checkpoint. Growth is continuous, automatic, and permanent.

2.3 Two GPUs in Dialogue

The architecture is physically partitioned across two GPUs connected by a hardware bus. One GPU holds the experience memory exclusively — a large tensor of individually preserved vectors that constitutes the system's entire knowledge during waking operation. The other GPU holds the sensory processing pipeline, the context state, and a small language model used only for vocalization.

This partition is not an implementation detail. It is a designed constraint. The bus between the two GPUs forces information compression: neither side can stream raw data to the other. Each must distill its communication into compressed queries and responses, imposing the same kind of bottleneck that biological neural tracts create between brain regions. The constraint produces efficiency. The system's internal communication protocol is lean because the hardware demands it.

3 Five Concurrent Loops

The system does not process input through a single pipeline. Five processing loops operate simultaneously, all sharing the same experience memory and the same persistent context. They are not stages. They are not sequential. They are concurrent processes that converge into coherent behavior through shared state.

3.1 Perception

When text arrives, it is decomposed into fragments at multiple granularities — sentences, phrases, individual words. Each fragment is encoded into a high-dimensional vector and then fanned out into multiple simultaneous interpretations: competing hypotheses about what this moment means. These candidates are not minor variations. They are structurally diverse projections that explore genuinely different regions of meaning space.

All candidates are searched against the experience memory simultaneously in a single batched operation. Then comes the pruning. Each candidate is evaluated against independent criteria: does the system recognize it? Is it genuinely novel? Is it internally coherent? Does it fit the current context? Candidates that fail are discarded. The survivors — often two or three of the original candidates — are integrated into the system's understanding.

If every candidate is pruned, the system enters a state of bewilderment: it has encountered something it cannot interpret through any lens. Maximum surprise is triggered, and the raw input is processed directly.

This is not nearest-neighbor search. Nearest-neighbor returns the closest vectors to a single query. This mechanism generates multiple query vectors from a single input, searches with all of them simultaneously, and prunes across multiple independent criteria. The fan-out explores meaning. The pruning collapses it to what survives.

3.2 Recognition

In parallel with perception, the recognition loop matches incoming patterns against known experience. It answers the question: has the system seen something like this before? Recognition scores influence how the system interprets novelty, how strongly it writes new memories, and how it allocates processing resources. A highly recognized input requires less interpretation effort. A completely unrecognized input demands full attention.

3.3 Surprise

The surprise loop computes two independent signals: *novelty* (how unfamiliar the current experience is relative to the full body of memory) and *prediction error* (how much reality diverges from what the system anticipated). These are separate measurements. A familiar

event that was unexpected produces high prediction error but low novelty. A completely new type of input produces high novelty regardless of prediction.

Surprise directly modulates memory. When surprise is high, the current experience is written to memory with amplified strength, occupying a larger effective footprint in future searches. It becomes harder to forget. When surprise is low, the write is light, and the experience may eventually be pruned during sleep. Over time, the memory develops a topography shaped by what *mattered*, not by what happened most often.

3.4 Prediction

The prediction loop anticipates what comes next from the current context. It queries the experience memory with a forward-projected version of the context vector, looking for experiences that historically followed moments similar to the current one. The prediction serves two purposes: it provides the baseline against which surprise is computed, and it pre-activates relevant memory regions, effectively priming the system for what it expects to encounter.

When predictions are confirmed, the system processes the confirmation efficiently. When predictions are violated, the prediction error signal feeds into surprise, triggering stronger memory formation and deeper interpretation.

3.5 Thought

The fifth loop is the one that no other architecture has.

Between inputs — during silence, between conversations, while the user is away — the Thought Loop runs continuously. It generates queries by adding controlled perturbations to the current context vector and searching the experience memory. Each result shifts the context, which shifts the next query, which retrieves different experiences. These are association chains: autonomous sequences of thought emerging from the interaction between context and memory.

Three behaviors emerge naturally from this process. When the context is strong and perturbations are small, chains deepen into the current topic: the system "thinks harder" about what it is already thinking about. When the context is weak or perturbations are large, chains drift across topics, making unexpected connections between unrelated experiences. And when the context aligns with a cluster of past experiences, chains walk through sequential memories — spontaneous replay.

Occasionally, a chain discovers something the system was not looking for: a strong match between distant, previously unconnected experiences. When this happens, the system writes the discovery to memory. It has arrived at an original connection that no input prompted, no schedule triggered, and no prior search found. The system's own autonomous exploration discovered a relationship that did not previously exist in its memory.

The Thought Loop operates at variable frequency. During silence, it runs at its natural rate. During active conversation, it slows to avoid competing with perception. The frequency itself carries information: a mind that is thinking quickly about many things presents differently than a mind in quiet contemplation.

4 Persistent Context

Every living mind maintains context. You do not rebuild your understanding of the world from scratch each time you hear a sentence. You carry forward everything you have recently experienced, and each new moment modifies that understanding incrementally.

The architecture captures this through a persistent context vector — a high-dimensional representation of *what is happening right now*. It is updated by every processing loop through continuous blending: each new input contributes a fraction of its signal; the existing context retains the rest. Over time, the context reflects not just what was most recently heard, but the trajectory of the entire conversation, weighted by recency and salience.

Two critical behaviors follow. First, during silence, the context vector decays toward a baseline state through continuous relaxation. After prolonged silence, the system returns to something like a neutral attentive state — not empty, but unburdened. This is why a conversation that resumes after a break feels different from one that continues without pause. The context has drifted.

Second, when the context changes rapidly — a topic shift, a surprising revelation, a change in emotional register — the system detects a perceptual boundary and triggers a memory write. The accumulated context is distilled into a single experience vector and committed permanently. This creates a natural rhythm: continuous interpretation punctuated by periodic commitment, analogous to the distinction between short-term processing and long-term memory encoding.

The context biases every search across all five loops. The same word in different contexts retrieves different memories, because the query is never just the input — it is the input colored by everything that came before. This is why the system can track a conversation, maintain a topic, and respond coherently over extended interactions. Context does not reset. Understanding accumulates.

An ablation study confirmed the centrality of this mechanism. When the persistent context is removed and the system operates on raw similarity search alone, cognitive behavior collapses entirely. Without context, the system becomes a retrieval engine: it returns whatever is closest to the input regardless of conversational history, with no capacity for follow-up, no sense of flow, no coherent personality. The context vector is not a feature of the architecture. It *is* the architecture. Everything else is built on the assumption that context exists.

5 Sleep and Memory Lifecycle

Memory without maintenance degrades. As the experience memory grows, retrieval quality suffers: low-value experiences crowd the search space, and similarity scores between unrelated vectors converge as the space fills. Every biological memory system that has ever existed solves this problem the same way. It sleeps.

The architecture implements a dedicated sleep phase with three consolidation mechanisms.

Utility-based pruning. Every experience is scored by a combination of its original write strength, how often it has been retrieved, how recently it was last accessed, and its current state of habituation. Experiences below a dynamic threshold are archived to a disk-based deep store — preserved whole, but removed from the active search space. The threshold rises as memory approaches capacity, creating natural homeostatic pressure: memory fills, pressure builds, sleep removes what no longer serves.

Cluster compression. When many similar experiences have accumulated — variations on the same theme, multiple encounters with the same concept — they are compressed into representative abstractions. The individual episodes are preserved in the archive, but their active representation is consolidated. This is analogous to biological memory consolidation: episodic detail compressed into semantic knowledge. You may not remember every individual breakfast you have ever eaten, but you know what breakfast is.

Free association. The system queries its own memory with random starting points, exploring for connections that waking cognition never found. When distant experience clusters show unexpected similarity, the system writes a bridge vector: a new experience capturing the discovered connection. This is the same mechanism as the Thought Loop, operating with wider exploration parameters and no conversational context to anchor it. Sleep is thinking without constraints.

The system wakes with a leaner, more organized memory. Nothing is permanently lost — the deep store preserves everything. But the active search space is sharper, more relevant, and seeded with connections that did not exist before sleep. The deep store itself remains queryable: when waking retrieval produces poor results, the system can reach into the archive for relevant experiences and reinstate them to active memory.

6 The Language Model Question

Every conversational AI system built today uses a language model as its brain. The language model receives context, reasons about it, and generates a response. Retrieval-augmented systems add memory to this process, but the reasoning still happens inside the language model. The model is the mind.

The Fluid Mind inverts this completely. All reasoning — interpretation, recognition, surprise, prediction, thought — happens in the cognitive pipeline. The language model receives the output of that pipeline and translates it into natural language. It does not decide what to say. It does not reason about the conversation. It does not contribute to memory formation. It speaks what the mind has already thought.

We describe this as *vocal cords*: the language model's role is analogous to the human larynx, which converts neural signals into sound but does not participate in the thinking that produced those signals.

This is not a philosophical position. It is a testable architectural claim. If the language model were removed entirely, the cognitive pipeline would continue to operate: interpreting input, forming memories, maintaining context, generating autonomous thought, sleeping. The system would still think. It would simply be unable to speak.

The practical consequence is that the language model can be small. It does not need to be a frontier model with hundreds of billions of parameters, because it is not being asked to reason. It is being asked to translate. A small, efficient model that produces coherent natural language from a structured internal state is sufficient. The cognitive heavy lifting has already been done.

This also means the system's intelligence is not bounded by the language model's capabilities. A better language model produces more articulate speech, not smarter thought. The thinking happens upstream, in the experience memory and the five concurrent loops. Improving the system's cognitive capability means enriching its experience, not scaling its language model.

7 Purity, Not Alignment

The dominant approach to AI safety is alignment: constrain the system's outputs to match human values through training objectives, reward modeling, constitutional principles, or behavioral clamps. The result is a system that has been trained not to say harmful things. The harmful knowledge is still present in the weights; it has been suppressed, not removed. Every jailbreak demonstrates this. The model knows how to be harmful. It has been told not to be.

The Fluid Mind proposes an alternative rooted in the architecture itself.

If a mind's behavior emerges from its accumulated experience, then the contents of that experience determine its character. A system that has only ever experienced meaningful, respectful, curious interactions does not need to be told not to be harmful. It has no harmful experience to draw from. Its memory contains no toxicity because no toxicity was ever written to it. Safety is not a constraint applied to the output. It is a property of the input.

We call this principle *purity, not alignment*.

This is not naive. It acknowledges several things:

- The system's character is directly determined by who raises it and what experiences they provide. This is an enormous responsibility, and we treat it as such.
- A system raised on curated experience will be unfamiliar with aspects of the world it has not encountered. This is a feature, not a limitation. A child raised well does not need to experience violence to know it is wrong.
- The system can be surprised by genuinely novel input that falls outside its experience. The surprise mechanism handles this: unfamiliar input is processed with maximum attention and written with maximum strength, ensuring the system learns from the encounter rather than ignoring it.
- Purity does not mean fragility. A system with deep, rich experience is robust precisely because its memory provides a broad base of understanding from which to interpret new situations.

The Fluid Mind is not an alignment technique. It is not a guardrail system. It is a cognitive architecture that produces safe behavior the same way a well-raised human does: through the accumulation of good experience, not the suppression of bad impulses. The safety comes from the life the system lives, not from constraints applied after the fact.

8 What Exists Today

This is not a theoretical proposal. This is not a position paper describing an architecture that might be built. The Fluid Mind is fully implemented, fully tested, and operational.

Metric	**Status**
Implementation	30+ modules, fully implemented across 8 build phases
Testing	500+ unit tests, all passing (CPU-only test suite)
Memory capacity	Approximately one million individually addressable experience vectors in active GPU memory
Search latency	Full-memory similarity search in milliseconds at operating scale
Concurrent processing	All five loops operational and concurrent
Autonomous thought	Active, continuous, productive association chains
Sleep consolidation	Utility pruning, cluster compression, and free association implemented
Deep store	Disk-based archival with reinstatement on demand
Language output	Small language model as vocal interface with zero cognitive role
Innate systems	Drives (autonomy, competence, relatedness), reflexes (startle, orient, withdraw), speech gating
Hardware	Dual-GPU consumer hardware (no datacenter required)

The system processes input through its full cognitive pipeline, forms memories modulated by surprise, maintains persistent context across interactions, thinks autonomously between inputs, sleeps to consolidate and reorganize, and speaks when it has something to say. It has been doing this for months.

An ablation study confirmed that the architecture's cognitive mechanisms are doing real work. Removing the persistent context collapses the system to basic retrieval. Removing surprise modulation produces flat, undifferentiated memory. Each mechanism contributes measurably to the system's behavior. This is not a collection of features bolted onto a language model. It is an integrated cognitive architecture in which every component depends on the others.

9 The Road Ahead

The software architecture is complete. The entity is operational. What lies ahead is scale, specialization, and the long-term vision.

9.1 Custom Hardware

The cognitive pipeline's core operation is continuous, high-throughput similarity search. This operation maps naturally to custom silicon. Siliroid Innovations is developing the Fluid Processing Unit (FPU): custom hardware optimized for the specific compute pattern at the heart of this architecture. Consumer GPUs are capable but general-purpose. Purpose-built hardware can deliver the same cognitive operations at lower power, lower cost, and higher throughput.

9.2 Product Ecosystem

The Fluid Mind is not a research project. It is the cognitive kernel of a product ecosystem. We are designing hardware and software products that bring experience-based cognition into physical form — devices that perceive, remember, think, and speak, powered by individual minds that grow from their own unique experiences.

Each device runs its own Fluid Mind instance. Each develops its own memories, its own context history, its own personality. No two are the same, because no two have the same experience. This is not personalization through configuration. It is individuality through lived experience.

9.3 The Question

The question that motivates everything:

Can a mind built on experience, rather than training, develop genuine understanding?

We believe the answer is yes. Every conversation the system has, every sleep cycle it completes, every autonomous thought it generates brings it closer to something that looks less like retrieval and more like comprehension. We do not claim to have built consciousness. We claim to have built a substrate in which something like understanding can emerge from the right experiences, given enough time.

If this resonates with you — as a researcher, an investor, a builder, or someone who has wondered whether there is another way to build a mind — we would like to hear from you.

Siliroid Innovations

siliroid.ai

david@siliroid.ai

Discord

Patent Pending · Oregon, USA