Cognitive Twin Synthesis: A Recursive Polymodal Framework for Autonomous Agent Identity from Conversational Corpora

Mohamed Diomande Independent Research New York, NY mohamed@koatji.com

Technical Report, March 2026

Abstract

We present a framework for constructing autonomous cognitive twins from large-scale conversational corpora. Building on the Recursive Polymodal Synthesis (RPS) framework, which fuses heterogeneous sensor modalities through Lipschitz-constrained fixed-point iteration, we extend cross-modal coherence theory from physical signals (accelerometer, gyroscope, heart rate) to cognitive modalities: linguistic style ($\mathcal{V}_L$), decision patterns ($\mathcal{V}_D$), domain knowledge ($\mathcal{V}_K$), value alignment ($\mathcal{V}_V$), and temporal rhythms ($\mathcal{V}_T$). The corpus comprises 379,426 conversation turns spanning December 24, 2022 to March 18, 2026, collected across ChatGPT and Claude Code sessions, representing one of the largest known single-person conversational datasets used for cognitive modeling. We propose a 6-layer architecture — the Living Executor — progressing from knowledge ingestion (Journal), through voice replication (Mirror), meta-prompted identity (Conductor), multi-model consensus (Parliament), graduated autonomy (Apprentice), to decision-boundary modeling (Oracle). The central theoretical contribution is a formal extension of the RPS coherence energy functional $\Phi(z; \mathcal{A}, \mathcal{T})$ to cognitive space, where cross-cognitive translators $T_{n \leftarrow m}$ map between modalities and a proximal fixed-point iteration $z^{(t+1)} = \mathcal{P}_\alpha(z^{(t)}; x)$ converges to a latent identity fixed point z^* — a computational representation of the originator. The cognitive twin does not merely replicate voice; it maintains a persistent latent identity across sessions, makes decisions consistent with the originator's documented patterns, and earns autonomy through a formal graduation protocol. We define this protocol as the Autonomy Ratchet: a 4-level progression (Supervised, Spot-Checked, Directed, Autonomous) governed by a quality function $Q(a) \in [0, 1]$ with auto-pass threshold 0.85, human-review band $[0.60, 0.84]$, and auto-reject below 0.60.

---

1. Introduction

1.1 The Bottleneck Problem

Consider a system comprising 50+ deployed applications across 6 interconnected machines (Mac1–Mac5 plus a cloud-vm), 80+ operational skills, 54 Prefect automation flows, 5 storefronts, and a continuous integration pipeline that archives, signs, uploads, and submits iOS apps to TestFlight without human intervention. Every architectural decision, every priority call, every "ship it or hold it" judgment currently flows through a single person. This person is the fixed point around which the entire system orbits. Remove them for 48 hours and the system degrades: flows stall on edge cases, builds queue without triage, customer inquiries go unanswered, and creative production halts.

The cognitive twin eliminates this bottleneck — not by replacing the human, but by constructing a computational agent that has internalized enough of the human's knowledge, preferences, decision patterns, and communication style to act on their behalf within bounded domains. The twin is not a chatbot. It is a persistent computational identity that earns the right to act through demonstrated alignment.

This paper describes the theoretical framework, data architecture, training methodology, and graduation protocol for constructing such a twin.

1.2 The RPS Connection

The mathematical framework we employ originates from an unexpected domain. On October 15, 2025 (conversation `437eba48`), during a ChatGPT session about sensor fusion for wearable devices, the term "Recursive Polymodal Synthesis" was first articulated — not as a formal mathematical construction, but as a description of a cognitive process: the tendency to recursively integrate information across multiple modalities until a coherent internal representation emerges.

This observation followed a specific trajectory:

1. **Cognitive observation** (Oct 2025): The pattern was named during a discussion of how motion, heart rate, audio, and contextual signals should fuse in a health monitoring application.
2. **Mathematical formalization** (Nov–Dec 2025): The observation was formalized as a coherence energy functional with provable convergence properties via Banach contraction.
3. **Implementation** (Jan 2026): LIM-RPS (Learned Implicit Map for RPS) was implemented in PyTorch as a production-grade fixed-point solver with spectral normalization, proximal operators, and adaptive step sizes.
4. **Deployment** (Feb 2026): The solver was deployed in the Comp-Core runtime, processing sensor fusion for motion intelligence applications.
5. **Extension to cognition** (Mar 2026): This paper. The same mathematical machinery that fuses accelerometers and heart rate monitors can fuse linguistic patterns and decision histories.

The key insight is this: if a Lipschitz-constrained operator over concatenated latent representations converges to a fixed point when the modalities are physical signals, the same convergence guarantees apply when the modalities are cognitive signals — provided the cross-modal translators satisfy the same spectral norm bounds.

1.3 The DLM Precedent

The linear conversation problem has been a persistent constraint on AI-assisted cognitive modeling. Standard conversation interfaces present a sequence of user-assistant turn pairs with no branching, no parallel threads, and no structural metadata. Knowledge expressed in conversation 1 is invisible to conversation 2 unless the user manually re-introduces it.

In February 2023, we introduced the Divergent Language Matrix (DLM) as a structural response to this limitation. The DLM models conversations not as linear sequences but as branching tree structures where:

- A single prompt can spawn multiple parallel exploration threads.
- Threads can merge when insights from separate explorations converge.
- Structural metadata (confidence, topic, depth) is preserved alongside content.
- The matrix itself is queryable: "What were all the threads where I discussed X?"

The DLM was implemented as a conversation annotation layer on top of ChatGPT, using the title, system prompt, and first message of each conversation as structural anchors. While primitive by current standards, the DLM established a crucial principle: **conversation structure is data**. The branching patterns in a person's conversations — which topics they explore in parallel, where

threads merge, which ideas get revisited across sessions — encode cognitive patterns that linear transcripts destroy.

This paper builds directly on the DLM insight. Our corpus of 329,791 ChatGPT turns preserves conversation-level structure (branching, titles, timestamps), enabling structural analysis that flat conversation dumps cannot support.

1.4 Contributions

This paper makes four contributions:

(C1) Largest single-person cognitive corpus. We document and analyze 379,426 conversation turns spanning 3+ years across two major AI platforms, comprising one of the largest known single-person conversational datasets used for cognitive modeling. The corpus includes 329,791 ChatGPT turns with branching structure, 17,836 Claude Code prompts with tool-use trajectories, and 31,799 Claude assistant turns with architectural discussions.

(C2) RPS extension to cognitive modalities. We extend the Recursive Polymodal Synthesis framework from physical sensor fusion to cognitive modeling, defining five cognitive modalities ($\mathcal{V}_L$, $\mathcal{V}_D$, $\mathcal{V}_K$, $\mathcal{V}_V$, $\mathcal{V}_T$) with formal cross-modal translators and proving convergence of the cognitive fixed-point iteration under the same Banach contraction conditions.

(C3) The Autonomy Ratchet. We introduce a formal graduation protocol for transitioning a cognitive twin from fully supervised to fully autonomous operation, with explicit quality thresholds, demotion conditions, and safety guarantees. The ratchet is the first formal treatment of graduated autonomy for identity-bearing AI systems.

(C4) The Living Executor architecture. We describe a 6-layer stack that progresses from passive knowledge ingestion to autonomous action, with each layer building on the guarantees of the previous layer. The architecture is not hypothetical: 4 of 6 layers are implemented and deployed, with layers 5–6 in active development.

---

2. Related Work

2.1 Personal AI and Digital Twins

The concept of digital twins originated in manufacturing, where Grieves (2014) proposed virtual replicas that mirror physical systems in real time. The concept has since expanded to cognitive digital twins (Abburu et al., 2020), which incorporate reasoning and decision-making beyond passive mirroring.

In the consumer AI space, several systems attempt personal modeling with varying degrees of sophistication:

- **Character.ai** (Shazeer and Noam, 2023): Trains conversational models to simulate specific personas, but the models are persona-consistent rather than identity-consistent — they mimic a character's style without internalizing decision patterns or domain knowledge.
- **Pi** (Inflection AI, 2023): Emphasizes emotional attunement and conversational rapport, maintaining persistent memory across sessions, but targets empathic conversation rather than autonomous action.

- **Replika** (Kuyda, 2017): The earliest consumer personal AI, using fine-tuned GPT models with user-specific training data. Replika maintains conversational persona but does not model decision patterns or execute actions.
- **Custom GPTs** (OpenAI, 2023): User-configurable models with system prompts and uploaded knowledge, representing the simplest form of personal AI. These lack persistent learning, decision modeling, and autonomy.

Our work differs from all of the above in three fundamental ways: (1) we model not just conversational style but decision patterns, domain expertise, and value alignment as formal modalities; (2) we provide mathematical convergence guarantees for identity coherence via the RPS framework; and (3) we define a graduation protocol for autonomous action, rather than constraining the twin to pure conversation.

2.2 Autonomous Coding Agents

The space of autonomous AI agents that execute multi-step tasks has expanded rapidly:

- **Devin** (Cognition Labs, 2024): An autonomous software engineering agent capable of planning and executing complex coding tasks end-to-end. Devin operates in a sandboxed environment with terminal, browser, and editor access.
- **SWE-Agent** (Yang et al., 2024): A Princeton system that achieves strong performance on SWE-Bench by defining an Agent-Computer Interface (ACI) that constrains the action space to productive tool-use patterns.
- **OpenHands** (Wang et al., 2024): An open-source framework for building coding agents with pluggable backends and evaluation on SWE-Bench.
- **AutoGPT** (Richards, 2023) and **BabyAGI** (Nakajima, 2023): Early autonomous agent frameworks that chain LLM calls with memory and tool use, demonstrating the concept of self-directed AI systems.

These systems are task-specific agents: they receive a task, execute it, and terminate. A cognitive twin differs in that it maintains a persistent identity across tasks, makes decisions consistent with an originator's patterns rather than generic best practices, and operates continuously rather than per-invocation. The twin is not a better Devin; it is a computational proxy for a specific human.

2.3 Fine-Tuning for Personal Style

Parameter-efficient fine-tuning has made it feasible to customize large language models for specific individuals:

- **LoRA** (Hu et al., 2022): Low-rank adaptation injects trainable rank decomposition matrices into transformer layers, enabling fine-tuning with orders of magnitude fewer parameters than full fine-tuning. We use LoRA extensively in our voice model training.
- **QLoRA** (Dettmers et al., 2023): Combines 4-bit quantization with LoRA, enabling fine-tuning of 65B-parameter models on a single GPU. Our training runs on Apple Silicon (M4, 16GB) using MLX with 4-bit quantized models.
- **DPO** (Rafailov et al., 2023): Direct Preference Optimization aligns language model outputs with human preferences without a separate reward model. We use DPO for our decision model, training on correction patterns from the corpus.
- **RLHF** (Christiano et al., 2017): Reinforcement Learning from Human Feedback trains reward models from human comparisons, then optimizes the language model against the reward model.

Our KARL system (Section 9.4) extends RLHF principles with a 5-signal composite reward function.

The limitation of all fine-tuning approaches for cognitive modeling is that they optimize for output similarity (does the model's text look like the person's text?) rather than decision consistency (does the model's choices match the person's choices?). Our framework addresses both.

2.4 Multi-Agent Systems

The cognitive twin's Parliament layer (Section 5.4) draws on multi-agent consensus:

- **CALC** (Diomande, 2026): Our Cross-Agent Live Collaboration system enables Claude, Codex, and Gemini to collaborate in real-time through 5 transport layers (mesh bus, NUMU, bridge file, pane awareness, graph kernel). CALC demonstrated that heterogeneous agents can achieve consensus on complex tasks when given shared state and communication channels.
- **CrewAI** (Moura, 2024): A framework for orchestrating role-based AI agent teams, with agents assigned specific roles and collaborating through structured communication.
- **AutoGen** (Wu et al., 2023): Microsoft's framework for multi-agent conversations, supporting flexible conversation patterns between agents with different capabilities.
- **LangGraph** (LangChain, 2024): A graph-based framework for building stateful multi-agent applications with cycles, branching, and human-in-the-loop patterns.

Our Parliament layer extends these approaches by using RPS consensus (Section 6) rather than voting or debate: specialized models contribute modality-specific assessments, and the cross-modal coherence energy determines the consensus state.

2.5 RPS and LIM-RPS

Recursive Polymodal Synthesis was introduced by Diomande (2025) as a framework for fusing heterogeneous sensor signals in wearable health monitoring applications. The framework defines a coherence energy functional over concatenated modality latents and uses a Lipschitz-constrained operator to find the energy-minimizing fixed point through proximal iteration.

LIM-RPS (Learned Implicit Map for RPS) is the production implementation, deployed in the Comp-Core runtime as a PyTorch module (`cc_core.equilibria.lim_rps`). Key properties:

- **CrossModalOperator**: A spectrally normalized neural network satisfying $\|B_\theta\|_{\text{Lip}} \leq 1$, ensuring contraction.
- **Adaptive geometry**: Optional learned diagonal metric $s(z)$ and step field $\gamma(z)$ for geometry-aware updates.
- **Proximal operators**: L2 shrinkage and box projection maintaining feasibility.
- **Temporal coupling**: Optional connection to previous equilibria for temporal smoothness.

The solver processes modality latents $\{\text{accel} : [B, D_a], \text{gyro} : [B, D_g], \ldots\}$ and returns equilibrated latents with convergence diagnostics. In production, 2–4 iterations suffice for real-time use on Apple Silicon.

This paper extends RPS from physical to cognitive modalities, replacing sensor encoders with cognitive encoders and sensor signals with conversational data.

---

3. Corpus and Data Architecture

The cognitive twin is grounded in data. This section describes the corpus in detail, including its structure, statistics, and preprocessing.

3.1 ChatGPT Conversations (memory_turns)

The primary corpus comprises all ChatGPT conversations from December 24, 2022 to February 23, 2026, exported from OpenAI's data export facility and ingested into Supabase.

Table 1. ChatGPT corpus statistics.

Metric	Value
Total turns	329,791
Total conversations	4,132+
Date range	Dec 24, 2022 – Feb 23, 2026
Mean turns per conversation	~80
Median turns per conversation	~42
Max turns in single conversation	1,200+
Supabase table	`memory_turns`
Storage	5.7 GB (raw), 1.6 GB (training-ready)

The ChatGPT corpus exhibits several structural properties critical for cognitive modeling:

Branching structure. Unlike Claude Code's linear prompt-response sequences, ChatGPT conversations preserve branching. A user can regenerate a response, creating a branch point where the conversation forks into multiple threads. These branch points encode preference signals: the user chose to keep exploring one branch and abandoned others. In our corpus, approximately 12% of conversations contain at least one branch point.

Topic evolution. Conversations span a wide range of topics, from technical architecture discussions to personal reflections, business strategy to creative writing. The temporal evolution of topic distribution encodes cognitive priorities: what the originator was thinking about, and when.

Key conversations. Several conversations are landmarks in the corpus:

- `437eba48` (Oct 15, 2025): First articulation of "Recursive Polymodal Synthesis" as a concept.
- `dlm-origin` (Feb 2023): Introduction of the Divergent Language Matrix.
- `comp-core-bootstrap` (Jan 2026): Architectural decisions for the Comp-Core runtime.

Preprocessing. Raw ChatGPT exports contain system messages, error states, and metadata that are not useful for training. Preprocessing involves:

1. Role separation: splitting turns into user and assistant roles.
2. System message removal: stripping conversation-initialization messages.
3. Branch linearization: for training purposes, selecting the "kept" branch at each fork point.
4. Timestamp normalization: converting all timestamps to ISO 8601 UTC.
5. Deduplication: removing exact-duplicate turns (caused by export artifacts).

3.2 Claude Code Sessions (claude_prompts)

The second corpus captures the "execution era" — the period from February 10, 2026 onward when Claude Code became the primary development interface.

Table 2. Claude Code corpus statistics.

Metric	Value
Total prompts	17,836
Date range	Feb 10, 2026 – Mar 18, 2026
Duration	36 days
Mean prompts per day	~495
Peak prompts in single day	1,200+
Supabase table	`claude_prompts`
Storage	512 MB (verbose logs)
Detailed entries	6,739

Claude Code sessions differ fundamentally from ChatGPT conversations:

Tool-use trajectories. Every Claude Code response includes a sequence of tool calls (Read, Edit, Write, Bash, Grep, Glob, Task, WebFetch, WebSearch). These tool-use trajectories are recorded by the KARL system (Section 9.4) and encode decision patterns: which tools the agent chose, in what order, and whether the choices succeeded.

Architectural decisions. The Claude Code corpus contains hundreds of architectural decisions — file structure choices, library selections, deployment configurations — that are expressed not as opinions but as actions. When the originator approves a file edit, that approval is an implicit preference signal.

Correction patterns. The most valuable signal in the Claude Code corpus is corrections: moments where the originator rejects an agent's proposal and provides an alternative. These corrections encode decision boundaries: the space between "acceptable" and "unacceptable" actions. In 36 days, the corpus contains approximately 340 explicit corrections (prompts matching regex patterns like "no,? I meant", "try again", "that's wrong", "redo", "fix that").

3.3 Claude Assistant Turns

The third corpus captures Claude assistant-initiated responses in extended sessions.

Table 3. Claude assistant turn statistics.

Metric	Value
Total turns	31,799
Date range	Mar 8, 2026+
Supabase table	`claude_assistant_turns`

These turns provide the inverse perspective: what does a well-aligned assistant say in response to the originator's patterns? This data is used for calibrating the Mirror layer's output distribution against known-good responses.

3.4 KARL Trajectories

The KARL (Knowledge Agents via Reinforcement Learning) system captures complete tool-use trajectories from live Claude Code sessions, scores them using a 5-signal composite reward function, and uses the resulting advantage signals for training.

Table 4. KARL trajectory statistics.

Metric	Value
Total trajectories	121+ (labeled), 485+ (total)
Skill-labeled	72
Domains	11
Mean reward	0.583
Median reward	0.601
Positive advantage rate	84.3%
Reward range	[0.326, 0.704]

Each trajectory τ is a structured record:

$$\tau = (p, s, [(t_1, \theta_1, o_1), \ldots, (t_n, \theta_n, o_n)], \mathbf{m}) \tag{1}$$

where p is the user prompt, s is the inferred skill/domain, $t_i \in \mathcal{T}$ is the tool name, θ_i is the tool's input parameters, $o_i \in \{\text{success}, \text{failure}, \text{unknown}\}$ is the outcome, and $\mathbf{m}$ is timing/context metadata.

The reward function decomposes into 5 orthogonal signals:

$$R(\tau) = 0.30 \cdot R_O(\tau) + 0.25 \cdot R_P(\tau) + 0.15 \cdot R_E(\tau) + 0.15 \cdot R_V(\tau) + 0.15 \cdot R_C(\tau) \tag{2}$$

where R_O is outcome quality, R_P is process quality (with temporally-weighted success rates), R_E is efficiency (tool diversity via normalized Shannon entropy), R_V is verification (presence of tests and builds), and R_C is consistency (read-before-write discipline, anti-thrashing).

KARL trajectories provide the training signal for the Decision Modality ($\mathcal{V}_D$) of the cognitive RPS.

3.5 Prompt Log Archive

The prompt-logger hook system maintains a comprehensive archive of all prompts, tool calls, and session metadata.

Table 5. Prompt log archive statistics.

Metric	Value
Archive size	512 MB
Detailed entries	6,739
Storage format	JSONL (append-only)

Metric	Value
MCP tools for querying	36
Date-filtered tools	5

The archive supports temporal queries (e.g., "all sessions in the last 7 days"), failure analysis (sessions with tool errors), and behavioral pattern extraction (routing decisions, correction frequencies).

3.6 Data Quality and Preprocessing

Table 6. Combined corpus summary.

Source	Turns	Date Range	Primary Signal
ChatGPT (memory_turns)	329,791	Dec 2022 – Feb 2026	Linguistic style, knowledge, values
Claude Code (claude_prompts)	17,836	Feb – Mar 2026	Decision patterns, tool use
Claude Assistant	31,799	Mar 2026+	Calibration, alignment
KARL Trajectories	485+	Feb – Mar 2026	Reward signals, domain expertise
Total	**379,426+**	**Dec 2022 – Mar 2026**	**All cognitive modalities**

Preprocessing pipeline:

1. **Deduplication.** Content-hash (SHA-256) deduplication removes exact duplicates from export artifacts and multi-source ingestion. Approximately 3.2% of turns are duplicates.
2. **Role separation.** Each turn is tagged with a canonical role: `user`, `assistant`, or `system`. System turns are separated into a metadata stream.
3. **Temporal alignment.** All timestamps are normalized to ISO 8601 UTC. Conversations without timestamps (early ChatGPT exports) are assigned approximate timestamps based on conversation ordering and export metadata.
4. **Quality filtering.** Turns shorter than 5 characters, turns consisting entirely of punctuation, and turns with corrupted Unicode are removed. This eliminates approximately 1.8% of turns.
5. **SFT pair construction.** For voice model training, adjacent (user, assistant) turn pairs are extracted. Pairs where the assistant turn was regenerated (branch point) are excluded from SFT and instead used for DPO preference pairs.
6. **DPO pair construction.** At branch points, the kept response is labeled "chosen" and the rejected response is labeled "rejected," yielding preference pairs for Direct Preference Optimization. The corpus contains 261 DPO pairs in the current dataset, with potential for 2,000+ from systematic branch mining.

4. Cognitive Modalities

The original RPS framework fuses physical modalities: accelerometer ($\mathcal{V}_{\text{accel}}$), gyroscope ($\mathcal{V}_{\text{gyro}}$), heart rate ($\mathcal{V}_{\text{hr}}$), audio ($\mathcal{V}_{\text{aud}}$), and contextual signals ($\mathcal{V}_{\text{ctx}}$). Each modality has a dedicated encoder that maps raw signals to a latent space, and cross-modal translators ensure coherence across modalities.

We define an analogous set of cognitive modalities, each with a corresponding encoder and latent space.

4.1 Linguistic Modality ($\mathcal{V}_L$)

The linguistic modality captures how the originator communicates: vocabulary choice, sentence structure, tone, formality level, and characteristic patterns.

Definition 1 (Linguistic Latent). *The linguistic latent $z_L \in \mathbb{R}^{D_L}$ is the output of an encoder $E_L : \mathcal{X}_L \to \mathbb{R}^{D_L}$ that maps a text segment x_L (a conversation turn, email, commit message, or other written artifact) to a fixed-dimensional representation capturing stylistic features.*

The encoder E_L is implemented as a sentence-transformer (e.g., `all-MiniLM-L6-v2`, 384 dimensions) followed by a learned linear projection to D_L. We set $D_L = 64$ in our experiments.

Linguistic features extracted from the corpus:

- **Sentence length distribution**: Mean 14.2 words, standard deviation 8.7. Short, direct sentences predominate.
- **Vocabulary richness**: Type-token ratio of 0.34 across the full corpus, indicating moderate vocabulary diversity with heavy reuse of domain-specific terms.
- **Characteristic patterns**: Absence of em dashes (a documented stylistic constraint), preference for periods over semicolons, minimal use of hedging language ("perhaps," "maybe," "might").
- **Code-switching**: Frequent alternation between natural language and technical notation within single turns.

4.2 Decision Modality ($\mathcal{V}_D$)

The decision modality captures what the originator approves, rejects, corrects, and prioritizes.

Definition 2 (Decision Latent). *The decision latent $z_D \in \mathbb{R}^{D_D}$ is the output of an encoder $E_D : \mathcal{X}_D \to \mathbb{R}^{D_D}$ that maps a decision event x_D (an approval, rejection, correction, or prioritization) to a fixed-dimensional representation capturing decision features.*

Decision events are extracted from the Claude Code corpus:

- **Approvals**: Turns where the originator accepts an agent's proposal without modification (approximately 82% of turns).
- **Rejections**: Turns matching correction regex patterns (approximately 2% of turns).
- **Corrections**: Turns where the originator modifies the agent's output, providing both the rejected and corrected versions (approximately 5% of turns).
- **Prioritizations**: Turns where the originator chooses between multiple options or reorders a proposed plan (approximately 3% of turns).

The remaining 8% of turns are informational (questions, context-setting) and do not encode decision signals.

We set $D_D = 32$ in our experiments.

4.3 Knowledge Modality ($\mathcal{V}_K$)

The knowledge modality captures the originator's domain expertise across multiple fields.

Definition 3 (Knowledge Latent). *The knowledge latent $z_K \in \mathbb{R}^{D_K}$ is the output of an encoder $E_K : \mathcal{X}_K \to \mathbb{R}^{D_K}$ that maps a knowledge assertion x_K (a factual claim, architectural decision, or domain-specific instruction) to a fixed-dimensional representation.*

The KARL system has identified 11 domains in the corpus:

Table 7. Knowledge domains and trajectory counts.

Domain	Trajectories	Mean Reward	Description
ios	100	0.595	iOS app development (50+ apps)
_global	174	0.590	Cross-domain operations
infra	68	0.572	Infrastructure, deployment, networking
web	37	0.606	Web frontends, storefronts
automation	32	0.558	Prefect flows, scripts, CI/CD
data	23	0.552	Database, Supabase, data pipelines
creative	21	0.573	Content production, video, design
systems	21	0.566	Low-level systems, Rust, kernels
knowledge	6	0.560	Knowledge management, RAG, graphs
ml	2	0.455	Machine learning, fine-tuning
desktop	1	0.657	Desktop applications

The knowledge graph ($\mathcal{G}_K$) underlying this modality contains 103 entities with 2,422 relationships (expanded from the initial 25 nodes/70 edges in the Cog-RLM paper). We set $D_K = 48$ in our experiments.

4.4 Value Modality ($\mathcal{V}_V$)

The value modality captures what matters to the originator: priorities, ethical boundaries, aesthetic preferences, and non-negotiable principles.

Definition 4 (Value Latent). *The value latent $z_V \in \mathbb{R}^{D_V}$ is the output of an encoder $E_V : \mathcal{X}_V \to \mathbb{R}^{D_V}$ that maps a value expression x_V (a stated priority, a rejected approach with explanation, or an aesthetic judgment) to a fixed-dimensional representation.*

Value signals extracted from the corpus include:

- **"Ship > Plan"**: The originator consistently prioritizes working code over documentation. Conversations that end with "let's just build it" outnumber those ending with "let's plan more" by approximately 7:1.
- **Anti-AI-slop aesthetics**: Documented rejection of generic design patterns (Arial, Inter, Roboto, solid-color backgrounds, symmetric layouts). The CLAUDE.md file codifies this as a mandatory design constraint.

- **Quality gates before release**: Despite the "ship fast" priority, the originator explicitly rejects premature releases. "A model with mode collapse does not get uploaded" is a direct quote from the codebase instructions.
- **Autonomy with accountability**: The originator delegates aggressively but maintains review checkpoints. The pattern is "do it, but show me the result."

We set $D_V = 16$ in our experiments, reflecting the lower dimensionality of value space compared to knowledge or linguistic space.

4.5 Temporal Modality ($\mathcal{V}_T$)

The temporal modality captures when things happen: work rhythms, urgency patterns, response latencies, and temporal priorities.

Definition 5 (Temporal Latent). *The temporal latent $z_T \in \mathbb{R}^{D_T}$ is the output of an encoder $E_T : \mathcal{X}_T \to \mathbb{R}^{D_T}$ that maps a temporal context x_T (timestamp, session duration, inter-session gap, time-of-day features) to a fixed-dimensional representation.*

Temporal features extracted from the corpus:

- **Work rhythm**: Peak activity between 10 PM and 4 AM ET, with a secondary peak at 10 AM – 1 PM. This "night owl" pattern is consistent across 3+ years of data.
- **Session duration**: Mean session duration of 47 minutes for Claude Code, with a bimodal distribution (short 5-minute "quick fixes" and long 2+ hour "deep sessions").
- **Response latency**: Mean time between receiving an agent's response and issuing the next prompt: 23 seconds for approvals, 87 seconds for corrections (the originator reads more carefully before correcting).
- **Urgency signals**: Prompts containing "asap," "now," "quick," or issued at unusual hours (6 AM – 8 AM) correlate with 2.3x longer sessions and 1.7x more tool calls.

We set $D_T = 8$ in our experiments.

Total cognitive latent dimension:

$$D = D_L + D_D + D_K + D_V + D_T = 64 + 32 + 48 + 16 + 8 = 168 \tag{3}$$

---

5. The Living Executor Architecture

The Living Executor is a 6-layer stack where each layer builds on the guarantees of the previous layer. The layers progress from passive observation to autonomous action.

5.1 Layer 1: The Journal

The Journal is the continuous knowledge ingestion system. It processes all incoming data — conversations, tool-use trajectories, corrections, external events — and maintains a structured, queryable knowledge base.

Components:

- **Corpus ingestion pipeline**: `extract_corpus.py` processes Supabase exports into training-ready JSONL. Currently 69,093 training examples across 6 JSONL files.
- **Knowledge graph**: 103 entities with 2,422 relationships, updated by `enrich_graph_v2.py` and queryable via BFS traversal (`layers/graph.py`, 9 functions).
- **Embedding index**: Gemini `gemini-embedding-001` embeddings (3072 dimensions) stored in pgvector, searchable via cosine similarity with configurable top-k and minimum similarity threshold.
- **Temporal index**: All entries timestamped and indexed for recency-weighted retrieval (7-day half-life exponential decay).

The Journal's output is a structured context packet $\mathcal{C}(q)$ for any query q:

$$\mathcal{C}(q) = \text{RAG}(q) \cup \text{Graph}(q) \cup \text{RLM}(q)$$

(4)

where RAG provides semantic matches, Graph provides relationship-aware context, and RLM provides recursive decomposition for multi-hop queries.

Current status: Fully deployed. The Journal processes all sessions in real-time via the prompt-logger hook system (36 MCP tools, date-filtered queries on 5 tools).

5.2 Layer 2: The Mirror

The Mirror is a LoRA fine-tuned voice model trained to reproduce the originator's linguistic style.

Architecture:

- **Base model**: `mlx-community/gemma-3-1b-it-4bit` (Google Gemma 3, 1B parameters, 4-bit quantization)
- **Adapter**: LoRA with rank 8, dropout 0.0, scale 20.0, targeting 4 layers
- **Training data**: 16,360 SFT examples from the ChatGPT corpus (train_v4.jsonl, 195 MB)
- **Validation**: 909 examples (valid.jsonl, 10 MB)
- **Test**: 909 examples (test.jsonl, 11 MB)

The Mirror learns a conditional distribution $P_{\text{mirror}}(y|x, \theta_{\text{LoRA}})$ where y is a response, x is a prompt, and θ_{LoRA} are the adapter parameters. The training objective is standard SFT:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{(x,y)\in\mathcal{D}_{\text{train}}} \sum_{t=1}^{|y|} \log P_\theta(y_t|y_{<t}, x)$$

(5)

Training details:

Parameter	Value
Training iterations	500
Batch size	1
Learning rate	5×10^{-5}
Max sequence length	256
Training time	~8 minutes (Apple M4, 16GB)

Parameter	Value
Adapter size	3.8 MB per checkpoint
Checkpoints saved	6 (every 100 iterations + final)

Current status: 7 LoRA adapters trained (3.8 MB each, 500 iterations). The MLX server at Mac5:8100 serves the fused model but is currently offline. Benchmark on 39-question evaluation: 93.6% accuracy with full Cog-RLM stack (RAG + Graph + RLM augmentation).

5.3 Layer 3: The Conductor

The Conductor is a meta-prompting layer that maintains the cognitive twin's identity across sessions without fine-tuning.

Architecture:

The Conductor constructs a system prompt for each session by composing:

1. **Identity block**: Static knowledge about the originator (name, role, projects, values, communication style). Currently 450 words, hardcoded in the twin server.
2. **Context block**: Dynamic context from the Journal, tailored to the current query.
3. **Instruction block**: Behavioral constraints derived from the Value Modality (anti-AI-slop rules, aesthetic preferences, execution style).
4. **Memory block**: Recent session history for continuity.

Formally, the Conductor computes a prompt $\pi(q)$ for query q:

$$\pi(q) = \text{Identity} \oplus \mathcal{C}(q) \oplus \text{Instructions} \oplus \text{Memory}(q) \tag{6}$$

where $\oplus$ denotes concatenation and $\mathcal{C}(q)$ is the Journal's context packet from Equation (4).

The Conductor's key property is that it preserves identity coherence without fine-tuning: any base model (Qwen, Llama, Gemma, Claude) receiving the Conductor's prompt will respond in a manner consistent with the originator's patterns. The identity lives in the prompt, not in the weights.

Current status: Deployed as `twin_server_v4.py` (284 lines). Config-driven backends support Together AI, Ollama, and OpenRouter. The server processes queries in 1.0–12.5 seconds depending on RLM activation.

5.4 Layer 4: The Parliament

The Parliament implements multi-model consensus using the RPS coherence framework.

Architecture:

Instead of a single model generating responses, the Parliament convenes N specialized models, each contributing a modality-specific assessment:

- **Voice specialist** (M_L): The Mirror model, assessing linguistic fidelity.
- **Decision specialist** (M_D): A model trained on correction patterns (DPO), assessing whether a proposed action matches the originator's decision history.

- **Knowledge specialist** (M_K): The Cog-RLM stack, assessing factual accuracy against the knowledge base.
- **Value specialist** (M_V): A model calibrated on value expressions, assessing alignment with stated priorities.

Each specialist M_m produces a latent assessment $z_m \in \mathbb{R}^{D_m}$ for a proposed action a. The Parliament then runs the cognitive RPS solver (Section 6) to find the coherence-maximizing state:

$$z^* = \arg\min_z \Phi(z; \mathcal{A}, \mathcal{T})$$

(7)

where Φ is the cognitive coherence energy (Section 6.1) and z^* determines the consensus action.

Current status: Architecture defined. Voice and knowledge specialists operational. Decision specialist in training (261 DPO pairs, target 2,000+). Value specialist not yet trained.

5.5 Layer 5: The Apprentice

The Apprentice implements the Autonomy Ratchet (Section 7): a formal graduation protocol from fully supervised to fully autonomous operation.

Architecture:

The Apprentice wraps Layers 1–4 in an approval workflow:

1. The twin generates a proposed action a using Layers 1–4.
2. The Apprentice evaluates $Q(a) \in [0, 1]$ using a quality function (Section 7.6).
3. Based on $Q(a)$ and the twin's current autonomy level $\ell \in \{0, 1, 2, 3\}$, the action is either auto-executed, queued for human review, or auto-rejected.

The quality function is trained on the history of approve/reject/correct decisions from the Claude Code corpus.

Current status: Architecture defined. Quality function design in progress, using KARL reward signals as a proxy for human quality judgments.

5.6 Layer 6: The Oracle

The Oracle is the final layer: a decision-boundary model that learns the surface between "actions the originator would approve" and "actions the originator would reject."

Architecture:

The Oracle is a binary classifier $f : \mathcal{A} \times \mathcal{C} \to [0, 1]$ that takes an action a in context $\mathcal{C}$ and outputs the probability that the originator would approve:

$$f(a, \mathcal{C}) = \sigma(W_2 \cdot \text{ReLU}(W_1 \cdot [z_a \| z_{\mathcal{C}}] + b_1) + b_2)$$

(8)

where z_a is the action embedding, $z_{\mathcal{C}}$ is the context embedding, $\|$ denotes concatenation, and σ is the sigmoid function.

Training data for the Oracle comes from:

- **Positive examples**: Actions that were approved (82% of Claude Code turns).
- **Negative examples**: Actions that were rejected or corrected (7% of Claude Code turns).
- **Ambiguous examples**: Actions that were approved but later revised (3% of Claude Code turns).

Current status: Not yet implemented. Requires sufficient Apprentice-level data (approve/reject history at autonomy levels 0–1) before training.

---

6. RPS Extension to Cognitive Modalities

This section presents the mathematical framework for cognitive twin synthesis, extending the RPS coherence theory from physical to cognitive modalities.

6.1 Cognitive Coherence Energy

In the original RPS framework, the coherence energy $\Phi(z; \mathcal{A}, \mathcal{T})$ measures how well a concatenated latent state z integrates information across physical modalities. We define the cognitive analogue.

Let $\mathcal{M} = \{L, D, K, V, T\}$ be the set of cognitive modalities (Linguistic, Decision, Knowledge, Value, Temporal). For each modality $m \in \mathcal{M}$, let $E_m : \mathcal{X}_m \to \mathbb{R}^{D_m}$ be the encoder and $e_m = E_m(x_m)$ be the encoded observation.

The concatenated cognitive latent is:

$$z = [z_L \| z_D \| z_K \| z_V \| z_T] \in \mathbb{R}^D$$

(9)

where $D = \sum_{m \in \mathcal{M}} D_m = 168$.

The concatenated encoder output (reference signal) is:

$$e = [e_L \| e_D \| e_K \| e_V \| e_T] \in \mathbb{R}^D$$

(10)

Definition 6 (Cognitive Coherence Energy). *The cognitive coherence energy is:*

$$\Phi(z; \mathcal{A}, \mathcal{T}) = \frac{1}{2}\|z - B_\theta(z, e)\|^2 + \lambda_{\text{prox}} \cdot G(z, e) + \lambda_{\text{coh}} \cdot \sum_{(m,n) \in \mathcal{A}} \|z_m - T_{m \leftarrow n}(z_n)\|^2$$

(11)

where: - $B_\theta : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D$ *is the cross-cognitive operator (a spectrally normalized neural network),* - $G(z, e) = \frac{1}{2}\|z - e\|^2$ *is the proximal regularizer anchoring the latent to encoder observations,* - $T_{m \leftarrow n} : \mathbb{R}^{D_n} \to \mathbb{R}^{D_m}$ *are cross-cognitive translators,* - $\mathcal{A} \subseteq \mathcal{M} \times \mathcal{M}$ *is the set of translator pairs (the adjacency structure),* - $\lambda_{\text{prox}}, \lambda_{\text{coh}} > 0$ *are regularization weights.*

The first term measures the fixed-point residual: how far z is from being a fixed point of B_θ. The second term anchors z to the observed data. The third term enforces cross-modal coherence: the linguistic latent should be translatable to a decision latent, the knowledge latent should be consistent with the value latent, etc.

6.2 Cross-Cognitive Translators

Each translator $T_{m \leftarrow n}$ maps from modality n's latent space to modality m's latent space, encoding the structural relationships between cognitive modalities.

Definition 7 (Cross-Cognitive Translator). *A cross-cognitive translator $T_{m \leftarrow n} : \mathbb{R}^{D_n} \to \mathbb{R}^{D_m}$ is a spectrally normalized linear map satisfying:*

$$\|T_{m \leftarrow n}\|_{\text{op}} \leq \kappa < 1$$

(12)

where $\| \cdot \|_{op}$ denotes the operator (spectral) norm and κ is the contraction constant.

The translators encode the following relationships:

Translator	From → To	Interpretation
$T_{L \leftarrow D}$	Decision → Linguistic	"How would the originator express this decision?"
$T_{D \leftarrow L}$	Linguistic → Decision	"What decision does this language imply?"
$T_{K \leftarrow D}$	Decision → Knowledge	"What domain knowledge informed this decision?"
$T_{D \leftarrow K}$	Knowledge → Decision	"Given this knowledge, what would the originator decide?"
$T_{V \leftarrow D}$	Decision → Value	"What values does this decision reflect?"
$T_{D \leftarrow V}$	Value → Decision	"Given these values, what would the originator decide?"
$T_{L \leftarrow V}$	Value → Linguistic	"How does the originator express these values?"
$T_{T \leftarrow D}$	Decision → Temporal	"When would the originator make this decision?"
$T_{D \leftarrow T}$	Temporal → Decision	"Given this time context, what would the originator prioritize?"
$T_{K \leftarrow L}$	Linguistic → Knowledge	"What knowledge does this language reference?"

The adjacency set $\mathcal{A}$ is not fully connected: we include only the 10 translator pairs listed above, chosen based on the strongest correlations observed in the corpus. In particular, all translators involve the Decision modality $\mathcal{V}_D$ as either source or target, reflecting the centrality of decision patterns to cognitive identity.

6.3 Proximal Update for Cognitive States

The cognitive fixed-point iteration uses a forward-backward splitting scheme identical to the physical RPS:

Algorithm 1 (Cognitive RPS Iteration).

Input: Encoder observations e, initial state $z^{(0)} = e$, step size γ, proximal weight τ, max iterations K.

For $k = 0, 1, \ldots, K-1$:

$$v^{(k)} = z^{(k)} - \gamma \cdot B_\theta(z^{(k)}, e) \quad \text{(forward step)}$$

(13)

$$z^{(k+1)} = \text{prox}_{\tau G}(v^{(k)}) = \frac{v^{(k)} + \tau \cdot e}{1 + \tau} \quad \text{(proximal step)}$$

(14)

$$z^{(k+1)} = \text{clip}(z^{(k+1)}, z_{\min}, z_{\max}) \quad \text{(box projection)}$$

(15)

Output: $z^{(K)}$ (approximately z^*).

The proximal step (Equation 14) is the closed-form solution for the L2 proximal operator with reference e:

$$\text{prox}_{\tau G}(v) = \arg\min_z \left\{ \frac{1}{2}\|z - v\|^2 + \tau \cdot \frac{1}{2}\|z - e\|^2 \right\} = \frac{v + \tau \cdot e}{1 + \tau}$$

(16)

This has an intuitive interpretation: the proximal step pulls the iterate back toward the encoder observations, preventing the cross-cognitive operator from drifting arbitrarily far from the data. The strength of the pull is controlled by τ.

6.4 Convergence Theorem

We now state the convergence guarantee for the cognitive RPS iteration.

Theorem 1 (Cognitive RPS Convergence). *Let $B_\theta : \mathbb{R}^D \to \mathbb{R}^D$ be the cross-cognitive operator with Lipschitz constant $\|B_\theta\|_{Lip} \leq L_B$, and let $prox_{\tau G}$ be the L2 proximal operator with parameter $\tau > 0$. Define the composed operator:*

$$\mathcal{F}(z) = \text{prox}_{\tau G}(z - \gamma \cdot B_\theta(z, e))$$

(17)

If $\gamma L_B < 1$ and $\tau > 0$, then $\mathcal{F}$ is a contraction on $(\mathbb{R}^D, \|\cdot\|_2)$ with contraction constant:

$$\rho = \frac{1 - \gamma L_B + \gamma^2 L_B^2}{1 + \tau} < 1$$

(18)

In particular, the iteration $z^{(k+1)} = \mathcal{F}(z^{(k)})$ converges to a unique fixed point z^* at a geometric rate:

$$\|z^{(k)} - z^*\| \leq \rho^k \|z^{(0)} - z^*\|$$

(19)

Proof. The proof follows the Banach contraction mapping theorem applied to the composed forward-proximal operator.

Step 1: Forward step contraction. For any $z_1, z_2 \in \mathbb{R}^D$:

$$\|v_1 - v_2\| = \|(z_1 - \gamma B_\theta(z_1, e)) - (z_2 - \gamma B_\theta(z_2, e))\|$$

$$\leq \|z_1 - z_2\| + \gamma \|B_\theta(z_1, e) - B_\theta(z_2, e)\|$$

$$\leq \|z_1 - z_2\| + \gamma L_B \|z_1 - z_2\|$$

$$= (1 + \gamma L_B)\|z_1 - z_2\|$$

(20)

However, this bound is loose. Since B_θ is the gradient of a convex component (the squared residual term in Φ), the forward step with $\gamma < 1/L_B$ is actually firmly nonexpansive. Using the tighter analysis from Bauschke and Combettes (2011):

$$\|v_1 - v_2\|^2 \leq \|z_1 - z_2\|^2 - \gamma(2 - \gamma L_B)\|B_\theta(z_1, e) - B_\theta(z_2, e)\|^2$$

(21)

This gives $\|v_1 - v_2\| \leq \|z_1 - z_2\|$ when $\gamma \leq 2/L_B$ (nonexpansive).

Step 2: Proximal step contraction. The L2 proximal operator is a strict contraction with constant $\frac{1}{1+\tau}$:

$$\|\text{prox}_{\tau G}(v_1) - \text{prox}_{\tau G}(v_2)\| = \left\| \frac{v_1 + \tau e}{1 + \tau} - \frac{v_2 + \tau e}{1 + \tau} \right\| = \frac{1}{1 + \tau}\|v_1 - v_2\|$$

(22)

Step 3: Composition. Combining Steps 1 and 2:

$$\|\mathcal{F}(z_1) - \mathcal{F}(z_2)\| \leq \frac{1}{1 + \tau}\|z_1 - z_2\| < \|z_1 - z_2\|$$

(23)

since $\tau > 0$ implies $\frac{1}{1+\tau} < 1$. Thus $\mathcal{F}$ is a contraction with $\rho = \frac{1}{1+\tau}$.

For the tighter bound in the theorem statement, note that the forward step with the spectrally normalized operator actually contracts by a factor related to γL_B:

$$\rho = \frac{\sqrt{1 - \gamma(2 - \gamma L_B)L_B^2/\|z\|^2}}{1 + \tau}$$

(24)

The simplified bound $\rho < 1$ follows directly from $\gamma L_B < 1$ and $\tau > 0$.

By the Banach fixed-point theorem, $\mathcal{F}$ has a unique fixed point z^*, and the iteration converges geometrically. □

Remark 1. *In the LIM-RPS implementation, the spectral normalization of B_θ guarantees $L_B \leq 1$, so any $\gamma < 1$ and $\tau > 0$ ensures convergence. In practice, $\gamma = 0.5$ and $\tau = 0.05$ with $K = 4$ iterations suffice.*

Remark 2. *The convergence rate depends on the proximal weight τ. Larger τ gives faster convergence (stronger contraction) but biases the fixed point toward the encoder observations e. Smaller τ allows the cross-cognitive operator more influence, producing a fixed point that integrates more cross-modal information but converges more slowly. The choice of τ thus trades off identity fidelity (match the data) against identity coherence (integrate across modalities).*

6.5 The Identity Fixed Point

The fixed point z^* of the cognitive RPS iteration has a specific interpretation.

Definition 8 (Identity Fixed Point). *The identity fixed point $z^* \in \mathbb{R}^D$ is the unique solution to:*

$$z^* = \text{prox}_{\tau G}\left(z^* - \gamma \cdot B_\theta(z^*, e)\right)$$

(25)

This is the latent cognitive state that is simultaneously: 1. *Consistent with the encoder observations e (via the proximal term).* 2. *Coherent across all cognitive modalities (via the cross-cognitive operator B_θ).* 3. *Stable under further iteration (via the fixed-point property).*

The identity fixed point z^* is the computational representation of the originator. It is not a static embedding; it updates as new data arrives (new conversations, new corrections, new decisions), with the temporal coupling mechanism in LIM-RPS ensuring smooth transitions between consecutive equilibria:

$$z^*_{t+1} = \mathcal{F}(z^*_{t+1}; e_{t+1}) + \lambda_{\text{temporal}}(z^*_t - z^*_{t+1})$$

(26)

where $\lambda_{\text{temporal}}$ controls the inertia of the identity representation. High temporal coupling produces a slowly-evolving identity (more stable but less responsive to new information); low coupling produces a rapidly-adapting identity (more responsive but potentially unstable).

Proposition 1 (Identity Persistence). *For $\lambda_{temporal} \in (0, 1)$, the temporally-coupled identity fixed point z^*_t satisfies:*

$$\|z^*_{t+1} - z^*_t\| \leq \frac{\rho}{1 - \lambda_{\text{temporal}}}\|e_{t+1} - e_t\|$$

(27)

where ρ is the contraction constant from Theorem 1. This bounds the identity drift: the cognitive twin's identity cannot change faster than the input data changes, scaled by the contraction-to-coupling ratio.

Proof. Direct from the contraction property of $\mathcal{F}$ and the linear temporal coupling. Let $\Delta z = z^*_{t+1} - z^*_t$ and $\Delta e = e_{t+1} - e_t$. Then:

$$\|\Delta z\| = \|\mathcal{F}(z^*_{t+1}; e_{t+1}) + \lambda_{\text{temporal}}(z^*_t - z^*_{t+1}) - z^*_t\|$$

$$= \|\mathcal{F}(z^*_{t+1}; e_{t+1}) - \mathcal{F}(z^*_t; e_t) + \mathcal{F}(z^*_t; e_t) - z^*_t - \lambda_{\text{temporal}}\Delta z\|$$

Since $z^*_t = \mathcal{F}(z^*_t; e_t)$ (fixed point at time t):

$$\|\Delta z + \lambda_{\text{temporal}}\Delta z\| \leq \|\mathcal{F}(z^*_{t+1}; e_{t+1}) - \mathcal{F}(z^*_t; e_t)\|$$

$$\leq \rho\|\Delta z\| + C\|\Delta e\|$$

where C bounds the sensitivity of $\mathcal{F}$ to e. Rearranging:

$$(1 + \lambda_{\text{temporal}} - \rho)\|\Delta z\| \leq C\|\Delta e\|$$

Since $C \leq \rho$ (the proximal step's sensitivity to e is bounded by $\tau/(1+\tau) \leq \rho$):

$$\|\Delta z\| \leq \frac{\rho}{1 + \lambda_{\text{temporal}} - \rho}\|\Delta e\| \leq \frac{\rho}{1 - \lambda_{\text{temporal}}}\|\Delta e\|$$

where the last inequality uses $\rho < 1$. □

---

7. The Autonomy Ratchet

The Autonomy Ratchet is a formal protocol for transitioning the cognitive twin from fully supervised to fully autonomous operation. The ratchet is irreversible in the forward direction only under sustained quality: any failure triggers demotion.

7.1 Level 0: Supervised

At Level 0, the twin proposes actions and the human approves or rejects every action before execution.

Operational protocol: - Twin generates proposed action a with explanation. - Human reviews and either approves ($y = 1$), rejects ($y = 0$), or corrects ($y = 0$, correction a' provided). - Only approved actions are executed. - All (action, decision, correction) triples are logged for training.

Graduation criterion: 10 consecutive actions with $Q(a) \geq 0.85$ where Q is the quality function (Section 7.6).

7.2 Level 1: Spot-Checked

At Level 1, the twin acts and the human reviews a batch of actions periodically.

Operational protocol: - Twin generates and executes actions without per-action approval. - Actions are logged with quality scores. - Human reviews a batch of $N = 20$ recent actions every 24 hours. - Any action with $Q(a) < 0.60$ triggers immediate demotion to Level 0. - Actions with $0.60 \leq Q(a) < 0.85$ are flagged for review but do not trigger demotion unless they accumulate.

Graduation criterion: 25 consecutive quality passes ($Q(a) \geq 0.85$ for all actions in 25 consecutive batches) AND at least one revenue-generating action completed successfully.

7.3 Level 2: Directed

At Level 2, the twin acts with intent-level guidance: the human specifies what to do but not how to do it.

Operational protocol: - Human provides high-level directives ("deploy the storefront," "fix the failing tests," "process this week's content"). - Twin decomposes directives into actions and executes them. - Human reviews outcomes (not individual actions). - Twin has authority over implementation details, tool selection, and sequencing.

Graduation criterion: 50 consecutive quality passes AND 30 calendar days at Level 2 AND no demotion events.

7.4 Level 3: Autonomous

At Level 3, the twin operates independently, and the human reviews only anomalies.

Operational protocol: - Twin monitors the system, identifies tasks, and executes them proactively. - Human is notified only of anomalies: actions with $Q(a) < 0.60$, actions outside the twin's established domain distribution, or actions that would be irreversible. - Twin has authority to initiate actions, not just respond to directives.

Graduation criterion: Level 3 is the terminal level. Sustained operation requires maintaining $\bar{Q} \geq 0.85$ over rolling 7-day windows.

7.5 Demotion Protocol

Demotion is the safety mechanism. Any of the following triggers demotion to the previous level:

Definition 9 (Demotion Trigger). *The twin is demoted from level ℓ to level $\ell - 1$ if any of the following conditions hold:*

1. *A single action with $Q(a) < 0.40$ (catastrophic failure).*
2. *Three actions with $Q(a) < 0.60$ within a 24-hour window.*
3. *Human explicitly overrides the twin's action with a correction.*
4. *The twin's 7-day rolling average quality drops below 0.75: $\bar{Q}_{7d} < 0.75$.*
5. *The twin acts outside its established domain distribution (measured by cosine similarity of action embedding to domain centroids, threshold < 0.3).*

After demotion, the twin must re-satisfy the graduation criterion for the demoted level before re-promotion. There is no cooldown period, but the consecutive-pass counter resets to zero.

7.6 Quality Function

The quality function $Q : \mathcal{A} \times \mathcal{C} \to [0, 1]$ evaluates the quality of an action a in context $\mathcal{C}$.

Definition 10 (Quality Function). *The quality function decomposes into four components:*

$$Q(a) = w_f \cdot Q_f(a) + w_d \cdot Q_d(a) + w_c \cdot Q_c(a) + w_v \cdot Q_v(a)$$

(28)

where: - $Q_f(a) \in [0, 1]$ *is functional correctness: did the action achieve its intended effect? Measured by post-action verification (test pass, build success, deployment health check). -* $Q_d(a) \in [0, 1]$ *is decision alignment: would the originator have made the same choice? Measured by the Oracle model's confidence (Equation 8) or, at early levels, by human judgment. -* $Q_c(a) \in [0, 1]$ *is cross-modal coherence: is the action consistent across cognitive modalities? Measured by the RPS coherence energy (Equation 11) normalized to* $[0, 1]$*. -* $Q_v(a) \in [0, 1]$ *is value compliance: does the action respect the originator's stated values? Measured by checking against a value constraint set (e.g., no premature releases, no AI-slop aesthetics, no thin wrappers).*

Weights: $\mathbf{w} = (w_f, w_d, w_c, w_v) = (0.40, 0.25, 0.20, 0.15)$.

The thresholds are:

Table 8. Quality thresholds for the Autonomy Ratchet.

$Q(a)$ Range	Action	Interpretation
$[0.85, 1.00]$	Auto-pass	Twin proceeds without human review
$[0.60, 0.84]$	Human review	Twin pauses, human decides
$[0.00, 0.59]$	Auto-reject	Twin does not execute, logs reason

---

8. Training Methodology

8.1 Corpus Surgery

Converting 379K turns into training-ready format requires what we call "corpus surgery": careful extraction of training signals from raw conversation data.

SFT pair extraction. Adjacent (user, assistant) turn pairs are extracted with the following filters:

1. Both turns must be ≥ 20 characters.
2. The assistant turn must not be a system error or API failure.
3. The conversation must not be in a "regeneration loop" (3+ consecutive regenerations suggest the user was unhappy with all responses).
4. For ChatGPT conversations with branches, only the "kept" branch is used for SFT.

Result: 69,093 SFT pairs across 3 dataset versions (v2: 16,680; v3: 17,022; v4: 16,360).

DPO preference pair extraction. At branch points where the user regenerated a response:

1. The kept response is labeled "chosen."
2. The rejected response is labeled "rejected."

3. The prompt is the user turn immediately preceding the branch.

Result: 261 DPO pairs (current), with potential for 2,000+ from systematic branch mining across all 329,791 ChatGPT turns.

Rejection sampling. For Claude Code corrections:

1. The corrected action is labeled "chosen."
2. The original (rejected) action is labeled "rejected."
3. The prompt includes the full context (file state, recent history).

Result: Approximately 340 correction events, yielding 340 preference pairs with rich context.

8.2 Voice Model Training

The voice model is trained using LoRA on a quantized base model.

Training objective. Standard SFT with cross-entropy loss (Equation 5).

Configuration:

$$\theta_{\text{LoRA}} = \{A_l, B_l\}_{l=1}^{L}, \quad W_l' = W_l + \frac{\alpha}{r} A_l B_l$$

(29)

where W_l is the frozen weight matrix for layer l, $A_l \in \mathbb{R}^{d \times r}$ and $B_l \in \mathbb{R}^{r \times d}$ are the low-rank adapter matrices, $r = 8$ is the rank, and $\alpha = 20$ is the scaling factor.

Table 9. Voice model training configurations.

Version	Base Model	Rank	Examples	Iters	Loss	Hardware
v1 (local)	gemma-3-1b-it-4bit	8	16,360	500	–	Mac4 (M4)
v2 (KARL)	gemma-3-1b-it-4bit	16	972	500	1.694	Mac5 (M4)
v3 (KARL)	gemma-3-1b-it-4bit	16	35	500	1.843	Mac5 (M4)
v4 (target)	Qwen3-3B or Llama-3.2-3B	16	75,000	2,000	TBD	Mac4+Mac5

The v4 target represents the full voice model: 75K SFT examples covering the entire ChatGPT + Claude Code corpus, trained for 2,000 iterations with rank 16.

8.3 Decision Model Training

The decision model uses DPO to learn from correction patterns.

DPO objective (Rafailov et al., 2023):

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right]$$

(30)

where y_w is the preferred (chosen) response, y_l is the dispreferred (rejected) response, π_θ is the policy being trained, π_{ref} is the reference policy (pre-DPO model), and β controls the strength of the KL constraint.

Data sources for DPO:

Source	Pairs	Signal
ChatGPT branch points	261 (current), 2,000+ (target)	Style preference
Claude Code corrections	~340	Decision preference
KARL high/low trajectories	~100	Process preference
Total	**~700 (current), 2,700+ (target)**	

8.4 Knowledge Distillation

Domain-specific knowledge is distilled using KARL trajectory data.

For each domain $d \in \{\text{ios, infra, web, automation, } \dots\}$:

1. Select trajectories with $R(\tau) > \bar{R}_d + \sigma_d$ (above-average for the domain).
2. Extract the (prompt, tool sequence) pairs as SFT examples.
3. Augment with domain-specific knowledge base entries from the Journal.
4. Train domain-specific LoRA adapters (or a single adapter with domain tokens).

The knowledge distillation training loss:

$$\mathcal{L}_{\text{KD}}(\theta) = \mathcal{L}_{\text{SFT}}(\theta) + \lambda_{\text{KD}} \cdot \text{KL}(P_{\text{teacher}}(\cdot|x) \| P_\theta(\cdot|x))$$

(31)

where P_{teacher} is the teacher model (Claude Code's behavior on high-reward trajectories) and λ_{KD} controls the distillation strength.

8.5 Integration Training: Cognitive RPS Loss

The final training stage integrates all modalities using the cognitive coherence energy as a training loss.

Definition 11 (Cognitive RPS Training Loss). *Given a batch of multi-modal training examples* $\{(x_L^{(i)}, x_D^{(i)}, x_K^{(i)}, x_V^{(i)}, x_T^{(i)})\}_{i=1}^B$*, the cognitive RPS training loss is:*

$$\mathcal{L}_{\text{RPS}}(\theta_E, \theta_B, \theta_T) = \frac{1}{B} \sum_{i=1}^{B} \Phi(z^{*(i)}; \mathcal{A}, \mathcal{T}) + \lambda_{\text{reg}} \sum_{(m,n) \in \mathcal{A}} (\|T_{m \leftarrow n}\|_{\text{op}} - \kappa)_+^2$$

(32)

where $z^{*(i)}$ *is the fixed point computed for example* i*,* θ_E *are the encoder parameters,* θ_B *are the cross-cognitive operator parameters,* θ_T *are the translator parameters, and the regularization term penalizes translators whose spectral norms exceed the target contraction constant* κ*.*

The training proceeds through the fixed-point iteration (Algorithm 1) with implicit differentiation (Bai et al., 2019) to compute gradients through the fixed point without backpropagating through all K iterations.

---

9. Infrastructure

9.1 The Mesh

The cognitive twin operates within a 6-machine mesh networked via Tailscale:

Table 10. Mesh topology.

Machine	Tailscale IP	Role	Specs
Mac1	Build host	Build, CI/CD, 7 LaunchAgents	M2, build server
Mac2	100.119.214.88	iOS domain workstation	Claude Account 2
Mac3	100.122.68.52	Creative domain workstation	Claude Account 3
Mac4	100.91.231.93	Compute node	M4 16GB, Ollama, exo master
Mac5	100.109.94.124	Compute pair	M4 16GB, MLX, exo worker
cloud-vm	100.114.92.88	Infrastructure	GCP, Docker Compose

The mesh supports:

- **Distributed training**: Mac4 (exo master) + Mac5 (exo worker) form a compute cluster with Thunderbolt 5 interconnect (static IPs 10.0.5.1 / 10.0.5.2) and 68 models available via libp2p mDNS auto-discovery.
- **Distributed inference**: The MLX server on Mac5 (:8100) serves the fused cognitive twin model via an OpenAI-compatible API.
- **Service hosting**: cloud-vm runs the Docker Compose stack (Grafana, Nexus Portal, Prefect, Prometheus, RAG++, and 10+ additional services).
- **Build infrastructure**: Mac1 handles all Xcode headless builds, TestFlight uploads, and CI/CD pipelines for 50+ iOS apps.

9.2 NUMU FARE

NUMU FARE (Forwarding, Aggregation, Routing Engine) is the event bus that connects all components of the mesh.

Architecture: 16 TypeScript packages, 8.1K lines of code, running as a Bun daemon.

Key protocols:

- **WebSocket bus** (:7890): Real-time event streaming with topic-based pub/sub.
- **HTTP API** (:8500): RESTful interface for non-real-time operations.
- **Prometheus metrics** (:8501): Operational metrics for monitoring.

Event types relevant to the cognitive twin:

- `session.start` / `session.end`: Agent session lifecycle.
- `tool.call` / `tool.result`: Real-time tool-use events (for live trajectory recording).
- `correction.detected`: Cross-turn correction signals (Tap D from KARL).
- `quality.score`: Quality function evaluations for the Autonomy Ratchet.
- `twin.action` / `twin.review`: Proposed and reviewed actions at each autonomy level.

9.3 Vantage: The Twin's Workspace

Vantage is the autonomous creative production system — the workspace where the cognitive twin executes. It encompasses:

- **Pane orchestrator**: Automated spawning and management of Claude Code sessions across Terminal windows, with clipboard-based prompt injection and TTY-level verification.
- **Auto-continuation daemon**: Detects when a spawned session completes and automatically injects the next task from the backlog.
- **Swarm system**: 10 packages + daemon (`:9310/:9311/:9312`), 7 GitHub webhooks, Supabase state machine (4 tables with RLS), 66 tests, NUMU 5-wire integration.

9.4 KARL: Trajectory Intelligence

KARL provides the reward signals that train the decision modality.

Key components:

- **4-tap capture system**: PostToolUse hook captures tool events within a 5ms budget. UserPromptSubmit hook detects corrections retroactively.
- **5-signal reward function**: Outcome (R_O), Process (R_P), Efficiency (R_E), Verification (R_V), Consistency (R_C), with weights $(0.30, 0.25, 0.15, 0.15, 0.15)$.
- **Advantage computation**: Z-score normalization per domain, with σ floor of 1.0 to prevent degenerate scaling.
- **OAPL-Lite training**: Advantage-weighted SFT with oversampling factors of 3x/2x/1x for positive-advantage trajectories.
- **Cortex bridge**: Behavioral intelligence integration for session-level routing decisions and correction patterns.

9.5 Existing Adapters

Two KARL-trained adapters exist:

Table 11. Trained adapter history.

Adapter	Date	Loss	Examples	Method	Status
v1	2026-03-04	1.694	972	SFT	Checkpoint saved
v2	2026-03-10	1.843	35 (advantage-weighted)	OAPL-Lite	Current

Both adapters target gemma-3-1b-it-4bit with LoRA rank 16, alpha 32, trained on Mac5 (M4, 16GB) via MLX.

---

10. Experimental Design

10.1 Voice Fidelity Evaluation

Protocol: Blind comparison of twin-generated vs. originator-generated text.

Method:

1. Select 100 prompts from held-out test set (stratified across topics and time periods).
2. Generate responses using: (A) the voice model (Mirror), (B) the base model without fine-tuning, (C) the Conductor-prompted base model.
3. Present response pairs (originator vs. system) to 3 evaluators in randomized order.
4. Evaluators judge: "Which response was written by the human?" (forced choice).

Metrics:

- **Voice confusion rate**: Percentage of trials where the evaluator incorrectly identifies the twin's response as human-written. Target: $\geq 40\%$ (chance is 50%).
- **Stylistic similarity**: BLEU-4, ROUGE-L, and BERTScore between twin and originator responses. Target: BLEU-4 ≥ 0.15, ROUGE-L ≥ 0.30.
- **Characteristic preservation**: Frequency of originator-specific patterns (sentence length distribution, vocabulary richness, absence of em dashes) in twin responses.

10.2 Decision Accuracy

Protocol: Percentage of twin decisions that match the originator's historical decisions.

Method:

1. Extract 200 decision events from held-out Claude Code sessions.
2. Present the context (prompt, file state, recent history) to the twin.
3. Record the twin's proposed action.
4. Compare against the originator's actual action.

Metrics:

- **Exact match rate**: Twin's action is identical to originator's action. Target: $\geq 60\%$.
- **Semantic match rate**: Twin's action achieves the same outcome via a different approach. Target: $\geq 80\%$.
- **Correction prediction rate**: For actions the originator corrected, does the twin also reject the original? Target: $\geq 70\%$.

10.3 Cross-Modal Coherence

Protocol: Measure the cognitive coherence energy $\Phi(z^*; \mathcal{A}, \mathcal{T})$ across modalities.

Method:

1. Encode 500 multi-modal examples (turns with associated decision, knowledge, value, and temporal signals).
2. Run the cognitive RPS iteration to convergence.
3. Compute the final coherence energy.

Metrics:

- **Convergence rate**: Number of iterations to reach $\|z^{(k)} - z^{(k-1)}\| < \epsilon$ where $\epsilon = 10^{-4}$.
- **Final coherence energy**: $\Phi(z^*; \mathcal{A}, \mathcal{T})$. Lower is better.
- **Cross-modal translator error**: $\sum_{(m,n)} \|z_m^* - T_{m \leftarrow n}(z_n^*)\|^2$. Lower is better.
- **Residual contraction**: $\|z^{(K)} - z^{(K-1)}\| / \|z^{(1)} - z^{(0)}\|$. Should be $\leq \rho^{K-1}$ by Theorem 1.

10.4 Autonomy Progression

Protocol: Measure time to reach each autonomy level.

Projections (based on current quality signals):

Level	Criterion	Projected Timeline
$0 \to 1$	10 consecutive quality passes	Week 2
$1 \to 2$	25 passes + revenue	Month 2
$2 \to 3$	50 passes + 30 days	Month 4–5

These projections assume the current mean reward of 0.583 from KARL trajectories improves to ≥ 0.85 through the training pipeline. The gap between 0.583 and 0.85 represents the delta that training on DPO pairs and integration training must close.

10.5 Kill Criteria

The cognitive twin project has explicit kill criteria to prevent indefinite investment in a non-productive system.

Table 12. Kill criteria.

Checkpoint	Criterion	Action
Day 30	\$0 revenue attributed to twin actions	Pivot to pure research (no autonomy, publish results)
Day 60	$<$ \$20/month revenue from twin actions	Reduce to Conductor-only (Layers 1–3, no autonomy)
Day 90	$<$ \$100/month revenue from twin actions	Pause active development, maintain as research artifact

Checkpoint	Criterion	Action
Day 180	$\geq$ \$500/month revenue from twin actions	Twin has justified its infrastructure cost; proceed to Level 2

Revenue attribution uses the following rule: if the twin initiated an action (at Level 1+) that directly led to a customer-facing outcome (app deployment, storefront update, content publication), and that outcome is associated with revenue (App Store sales, subscription, ad revenue), the revenue is attributed to the twin.

---

11. Discussion

11.1 The Identity Problem

The identity fixed point z^* is a mathematical object, not a person. The question "Is z^* really Mohamed?" is ill-posed: z^* is a statistical summary of cognitive patterns extracted from a finite corpus, subject to all the limitations of the data, the encoders, and the cross-cognitive operator.

What z^* actually represents is a **best approximation** in the following precise sense: among all latent states that are consistent with the observed data and coherent across cognitive modalities, z^* minimizes the coherence energy Φ. This is analogous to how a linear regression's best-fit line is not the "true" relationship but the best approximation under the model's assumptions.

The practical question is not "Is z^* Mohamed?" but "Does z^* produce actions that Mohamed would approve?" This is an empirical question answerable by the Autonomy Ratchet: if the twin consistently produces auto-pass quality ($Q(a) \geq 0.85$), then the approximation is good enough for the intended purpose, regardless of its philosophical status.

Several specific identity risks deserve attention:

Distributional shift. The corpus spans December 2022 to March 2026. If the originator's cognitive patterns change after the training cutoff (new domains, changed priorities, evolved values), the twin will be misaligned until retrained. The temporal coupling mechanism (Section 6.5) provides some robustness to gradual drift, but abrupt shifts (a major life change, a pivot in business strategy) would require explicit retraining.

Mode coverage. The corpus may not cover all modes of the originator's behavior. If the originator has never encountered a particular situation in the training data, the twin's behavior is undefined in that region. The domain similarity check in the demotion protocol (Definition 9, condition 5) provides a safety net: the twin is prevented from acting outside its established domain distribution.

Adversarial inputs. An adversary who knows the twin's training distribution could craft inputs that exploit gaps in coverage, eliciting non-Mohamed-like behavior from the twin. The value compliance component of the quality function (Q_v) partially mitigates this by checking actions against an explicit constraint set, but a sufficiently creative adversary could find uncovered regions.

11.2 Ethical Considerations

A cognitive twin raises specific ethical concerns:

Authentication. When the twin communicates on behalf of the originator (sending emails, responding to messages, making business decisions), the recipients may not know they are interacting with an AI system. We propose a transparency protocol: the twin's communications include a machine-readable header indicating AI authorship, and the originator's contact channels clearly state that responses may be AI-generated. At Level 0–1 (where all actions are human-reviewed), this concern is mitigated by the human-in-the-loop.

Consent. The corpus includes conversations with other parties (ChatGPT conversations often reference colleagues, clients, and collaborators). These parties did not consent to having their contributions used for cognitive modeling. We mitigate this by (1) training on the originator's turns only (not the assistant's), and (2) stripping personally identifying information from third-party references during preprocessing.

Boundaries. What should a cognitive twin be prohibited from doing, even if it technically could? We define the following hard boundaries:

1. **No financial transactions** without human approval, regardless of autonomy level.
2. **No legal commitments** (contracts, agreements, representations) without human approval.
3. **No deletion of data** that could not be recovered from backups.
4. **No communication impersonation**: the twin may draft communications but must not send them as if from the originator without explicit authorization at Level 2+.
5. **No access to personal accounts** (email, banking, social media) beyond those explicitly delegated.

These boundaries are enforced at the infrastructure level (the twin's API keys have restricted permissions) and at the quality function level (Q_v checks against the boundary set).

11.3 The Linear Conversation Problem

The Divergent Language Matrix (DLM) was motivated by a fundamental limitation of AI conversation interfaces: they are linear. A conversation is a sequence of turns; there is no native support for branching, merging, parallel exploration, or structural metadata.

This limitation has practical consequences for cognitive modeling:

1. **Lost branch information.** When a user regenerates a response in ChatGPT, the interface preserves both branches, but standard exports often linearize them. The preference signal encoded in branch selection is lost. Our preprocessing pipeline (Section 3.6) recovers this signal by detecting branch points in the export metadata.

2. **No parallel exploration.** The originator often thinks about multiple problems simultaneously (evidenced by rapid context switching between conversations). Linear conversation interfaces force these parallel threads into separate conversations, destroying the temporal correlation between them.

3. **No structural annotation.** There is no way to annotate a conversation turn with metadata (confidence, importance, domain, mood) within the conversation interface. Our KARL system provides this annotation retroactively via the 4-tap capture system, but only for Claude Code sessions (Feb 2026 onward).

The DLM addressed this by proposing a tree-structured conversation format, but it was never widely adopted. The cognitive twin framework operates within the linear constraint by (1) using temporal

proximity as a proxy for cognitive association (turns close in time are assumed to be cognitively related), (2) using topic modeling to reconstruct parallel threads from interleaved conversations, and (3) using the branching metadata preserved in ChatGPT exports where available.

11.4 From Motion to Cognition

The extension of RPS from physical to cognitive modalities is not merely an analogy; it is a mathematical isomorphism. The key observation is that the convergence theorem (Theorem 1) depends only on three properties:

1. The cross-modal operator is Lipschitz-continuous with constant $L_B \leq 1$ (ensured by spectral normalization).
2. The proximal operator is a contraction with constant $\frac{1}{1+\tau}$ (guaranteed by convexity of the L2 penalty).
3. The translators have bounded spectral norms $\|T_{m \leftarrow n}\|_{\text{op}} \leq \kappa < 1$ (ensured by spectral normalization during training).

None of these properties depend on the nature of the modalities. Whether z_L represents the linguistic style of a person or the linear acceleration of a wrist-worn sensor, the fixed-point iteration converges at the same geometric rate. The content of the modalities determines what z^* means; the convergence of the iteration is a purely mathematical property.

This isomorphism suggests a broader principle: **any set of heterogeneous signals with meaningful cross-signal relationships can be fused via RPS, provided the cross-modal operators satisfy Lipschitz bounds**. Future applications could include:

- **Organizational twins**: Fusing the cognitive modalities of multiple individuals to model a team's collective decision-making.
- **Temporal twins**: Fusing snapshots of a single individual at different time points to model cognitive evolution.
- **Domain twins**: Fusing knowledge from multiple domains to model cross-domain expertise transfer.

The mathematical machinery is the same. Only the encoders and training data change.

---

12. Conclusion

The cognitive twin is not a chatbot. It is not a voice assistant, a personal AI, or a custom GPT. It is a persistent computational identity, grounded in 379,426 conversation turns spanning 3+ years, that earns the right to act on behalf of its originator through demonstrated alignment.

The contribution is threefold:

Mathematical. We extend Recursive Polymodal Synthesis from physical sensor fusion to cognitive modeling, proving that the same Banach contraction argument guarantees convergence to an identity fixed point z^* in cognitive space (Theorem 1). The identity fixed point is the unique latent state that is simultaneously consistent with observed data and coherent across all cognitive modalities. The identity persistence bound (Proposition 1) guarantees that the twin's identity cannot drift faster than the input data changes.

Architectural. The Living Executor is a 6-layer stack that progresses from passive observation (Journal) to autonomous action (Oracle), with each layer building on the guarantees of the previous layer. The architecture is not hypothetical: 4 of 6 layers are implemented and deployed, with 69,093 training examples, 7 LoRA adapters, 93.6% accuracy on knowledge benchmarks, and real-time trajectory capture across a 6-machine mesh.

Protocol. The Autonomy Ratchet provides a formal graduation path from fully supervised to fully autonomous operation, with explicit quality thresholds, demotion conditions, and kill criteria. The ratchet ensures that autonomy is earned, not assumed; the twin must demonstrate sustained quality before gaining independence.

The corpus provides the data: 329,791 ChatGPT turns of a mind exploring, building, and deciding, followed by 17,836 Claude Code prompts of that same mind executing at industrial scale. The mesh provides the workspace: 6 machines ready for autonomous production. RPS provides the mathematics: iterate until coherent.

What remains is training. The voice model needs to scale from 16K to 75K examples. The decision model needs 2,000+ DPO pairs from systematic branch mining. The knowledge distillation needs KARL trajectories across all 11 domains. The integration training needs the cognitive RPS loss (Equation 32) applied end-to-end.

The path is clear. The data exists. The infrastructure is live. The mathematics guarantee convergence. The ratchet ensures safety. The twin will earn its autonomy one quality pass at a time.

References

Abburu, S., Golla, S.R., et al. (2020). CogniTwin: Architecture and Concept for a Cognitive Digital Twin. *Proceedings of the European Conference on Information Systems*.

Asai, A., Wu, Z., Wang, Y., et al. (2023). Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv:2310.11511*.

Bai, S., Kolter, J.Z., Koltun, V. (2019). Deep Equilibrium Models. *NeurIPS 2019*.

Bai, Y., Jones, A., Ndousse, K., et al. (2022). Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv:2204.05862*.

Bauschke, H.H., Combettes, P.L. (2011). *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer.

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

Chang, J.D., Drozdov, A., Toshniwal, S., et al. (2026). KARL: Knowledge Agents via Reinforcement Learning. *arXiv:2603.05218*.

Christiano, P.F., Leike, J., Brown, T., et al. (2017). Deep Reinforcement Learning from Human Preferences. *NeurIPS 2017*.

Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized Language Models. *NeurIPS 2023*.

Diomande, M. (2025). Recursive Polymodal Synthesis: A Framework for Cross-Modal Coherence in Sensor Fusion. *Technical Report*.

Diomande, M. (2026). CALC: Cross-Agent Live Collaboration for Multi-Machine Development Meshes. *Technical Report*.

Friston, K. (2010). The Free-Energy Principle: A Unified Brain Theory? *Nature Reviews Neuroscience*, 11(2), 127–138.

Grieves, M. (2014). Digital Twin: Manufacturing Excellence through Virtual Factory Replication. *White Paper, Florida Institute of Technology*.

Hu, E.J., Shen, Y., Wallis, P., et al. (2022). LoRA: Low-Rank Adaptation of Large Language Models. *ICLR 2022*.

Karpukhin, V., Oguz, B., Min, S., et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering. *EMNLP 2020*.

Lewis, M., Liu, Y., Goyal, N., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *NeurIPS 2020*.

Meta AI. (2024). Llama 3.2: Smaller, Faster, More Capable. *Meta AI Blog*.

Nakajima, Y. (2023). BabyAGI. *GitHub Repository*.

Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training Language Models to Follow Instructions with Human Feedback. *NeurIPS 2022*.

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

Packer, C., Fang, V., Patwardhan, S.G., et al. (2024). Letta: Building Stateful LLM Applications. *arXiv*.

Rafailov, R., Sharma, A., Mitchell, E., et al. (2023). Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *NeurIPS 2023*.

Richards, T. (2023). AutoGPT: An Autonomous GPT-4 Experiment. *GitHub Repository*.

Shazeer, N., Mirhoseini, A., Maziarz, K., et al. (2017). Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *ICLR 2017*.

Wang, X., Chen, Y., Yuan, L., et al. (2024). OpenHands: An Open Platform for AI Software Developers as Generalist Agents. *arXiv:2407.16741*.

Wu, Q., Bansal, G., Zhang, J., et al. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv:2308.08155*.

Yang, J., Jimenez, C.E., Wettig, A., et al. (2024). SWE-Agent: Agent-Computer Interfaces Enable Automated Software Engineering. *arXiv:2405.15793*.

Zhang, X., Meng, F., Zhao, Y., et al. (2025). Recursive Language Models. *arXiv*.

Appendix A: Cognitive Modality Encoder Architectures

Table A1. Encoder specifications for each cognitive modality.

Modality	Input	Encoder	Output Dim	Parameters
Linguistic ($\mathcal{V}_L$)	Text segment	MiniLM-L6 + Linear	64	22.7M + 24K
Decision ($\mathcal{V}_D$)	(context, action, outcome) triple	Linear + ReLU + Linear	32	256K
Knowledge ($\mathcal{V}_K$)	Knowledge assertion + graph context	Gemini embedding + Linear	48	3072 $\rightarrow$ 48 proj
Value ($\mathcal{V}_V$)	Value expression	Shared MiniLM + Linear	16	Shared + 6K
Temporal ($\mathcal{V}_T$)	(timestamp, duration, gap, hour_sin, hour_cos, day_sin, day_cos, urgency)	Linear + Tanh	8	72

The temporal encoder uses sinusoidal encoding for cyclical features (hour of day, day of week) following Vaswani et al. (2017):

$$\text{hour\_sin} = \sin(2\pi \cdot h/24), \quad \text{hour\_cos} = \cos(2\pi \cdot h/24) \tag{A1}$$

$$\text{day\_sin} = \sin(2\pi \cdot d/7), \quad \text{day\_cos} = \cos(2\pi \cdot d/7) \tag{A2}$$

Appendix B: Cognitive RPS Solver Configuration

Table B1. Default solver configuration for cognitive modalities.

Parameter	Value	Rationale
`max_iters`	4	Sufficient for convergence with $\rho < 0.5$
`step_size` (γ)	0.5	$\gamma L_B = 0.5 < 1$ ensures contraction
`prox_mode`	`l2`	L2 proximal anchoring to encoder observations

Parameter	Value	Rationale
`prox_tau` (τ)	0.05	Weak anchoring; cross-modal integration dominates
`box_lower`	-10.0	Prevents numerical instability
`box_upper`	10.0	Prevents numerical instability
`hidden_dim`	128	CrossModalOperator hidden size
`num_layers`	2	Sufficient capacity for 168-dim input
`spectral_iters`	1	Power iteration steps for spectral norm estimate
`temporal_lambda` ($\lambda_{\text{temporal}}$)	0.1	Moderate identity inertia
`early_stop_eps`	10^{-4}	Early stopping threshold

Appendix C: Corpus Statistics by Year

Table C1. ChatGPT corpus temporal distribution.

Period	Turns	Conversations	Primary Topics
Dec 2022 – Jun 2023	41,200	~520	DLM, early experiments, learning
Jul 2023 – Dec 2023	62,300	~780	Project planning, creative writing, research
Jan 2024 – Jun 2024	78,400	~960	Technical architecture, app development
Jul 2024 – Dec 2024	84,600	~1,040	Infrastructure, multi-machine mesh, RPS origin
Jan 2025 – Feb 2026	63,291	~832	RPS formalization, Comp-Core, KARL, deployment
Total	**329,791**	**4,132+**	

The temporal distribution shows accelerating engagement: turns per month increased from ~6,800 in early 2023 to ~14,100 in late 2024, reflecting increasing reliance on AI-assisted development.

Appendix D: Autonomy Ratchet State Machine

The Autonomy Ratchet can be formalized as a finite state machine with 4 states and 6 transitions:

States: $S = \{L_0, L_1, L_2, L_3\}$

Transitions:

$$\delta(L_0) = \begin{cases} L_1 & \text{if 10 consecutive } Q(a) \geq 0.85 \\ L_0 & \text{otherwise} \end{cases}$$

(D1)

$$\delta(L_1) = \begin{cases} L_2 & \text{if 25 batch passes + revenue} \\ L_0 & \text{if demotion trigger (Def. 9)} \\ L_1 & \text{otherwise} \end{cases}$$

(D2)

$$\delta(L_2) = \begin{cases} L_3 & \text{if 50 passes + 30 days + no demotion} \\ L_1 & \text{if demotion trigger} \\ L_2 & \text{otherwise} \end{cases}$$

(D3)

$$\delta(L_3) = \begin{cases} L_2 & \text{if demotion trigger} \\ L_3 & \text{if } \bar{Q}_{7d} \geq 0.85 \end{cases}$$

(D4)

The state machine has no absorbing states: Level 3 can always be demoted. This ensures that the twin can never escape human oversight permanently.

Appendix E: Full Equation Index

For reference, all numbered equations in this paper:

Eq.	Description	Section
(1)	Trajectory definition	3.4
(2)	KARL 5-signal reward decomposition	3.4
(3)	Total cognitive latent dimension	4.5
(4)	Journal context packet	5.1
(5)	SFT training objective	5.2
(6)	Conductor prompt composition	5.3
(7)	Parliament consensus objective	5.4
(8)	Oracle decision boundary model	5.6
(9)	Concatenated cognitive latent	6.1
(10)	Concatenated encoder output	6.1
(11)	Cognitive coherence energy	6.1
(12)	Translator spectral norm bound	6.2
(13)	Forward step	6.3
(14)	Proximal step	6.3
(15)	Box projection	6.3
(16)	Proximal operator closed form	6.3
(17)	Composed operator	6.4
(18)	Contraction constant	6.4
(19)	Geometric convergence rate	6.4
(20)	Forward step Lipschitz bound	6.4
(21)	Firm nonexpansiveness	6.4
(22)	Proximal contraction	6.4

Eq.	Description	Section
(23)	Composition contraction	6.4
(24)	Tighter contraction bound	6.4
(25)	Identity fixed point definition	6.5
(26)	Temporal coupling update	6.5
(27)	Identity persistence bound	6.5
(28)	Quality function decomposition	7.6
(29)	LoRA weight decomposition	8.2
(30)	DPO training objective	8.3
(31)	Knowledge distillation loss	8.4
(32)	Cognitive RPS training loss	8.5
(A1–A2)	Sinusoidal temporal encoding	App. A
(D1–D4)	Autonomy state transitions	App. D