Subsumption Pattern Learning: A Formal Framework for Self-Distilling Swarm Intelligence Through Shared Collective Memory

Pamela Cuce *

Posted Date: 30 January 2026

doi: 10.20944/preprints202601.2348.v1

Keywords: swarm intelligence; shared collective memory; multi-agent learning; subsumption architecture; inhibition signals; pattern distillation; foundation model economics; LLM cascades

 doi:10.20944/preprints202601.2348.v1

Article

Subsumption Pattern Learning: A Formal Framework for Self-Distilling Swarm Intelligence Through Shared Collective Memory

Pamela Cuce

Tufts University, USA; pamela.cuce@tufts.edu

Abstract

We introduce **Subsumption Pattern Learning (SPL)**, a hierarchical multi-agent framework that transforms collections of autonomous AI agents into a self-distilling swarm intelligence through shared collective memory. SPL adapts Brooks' subsumption architecture from behavioral robotics to foundation model economics, implementing a formally-defined three-layer hierarchy (Reactive, Tactical, Deliberative) where learned patterns are distilled into a centralized Shared State via explicit inhibition signals. We provide a complete mathematical formalization of the pattern distillation process, defining state transitions from deliberative reasoning to tactical reflexes through confidence-bounded suppression logic. Our framework unifies three previously disparate research streams: subsumption control from robotics, social learning theory from cognitive science, and swarm intelligence from distributed systems. We present rigorous empirical evaluation on a benchmark of 100,000 heterogeneous enterprise tasks, demonstrating 5–15× cost reduction per agent with an additional 40% reduction in foundation model escalations across coordinated multi-agent networks. Ablation studies confirm that cost savings preserve accuracy within 1.3% of baseline. We formalize the *intelligence compounding* phenomenon, proving that collective competency grows logarithmically with processed requests under mild assumptions. SPL provides a principled path toward AI systems that grow more intelligent with every transaction while maintaining robustness through decentralized resilience.

Keywords: swarm intelligence; shared collective memory; multi-agent learning; subsumption architecture; inhibition signals; pattern distillation; foundation model economics; LLM cascades

---

1. Introduction

1.1. Motivation: The Isolated Agent Problem

Large language models (LLMs) have become foundational infrastructure for autonomous AI agent systems, powering applications from customer service automation to software engineering assistants [15,19]. Yet enterprise deployments reveal a fundamental architectural limitation: modern LLM-based agents operate as *isolated computational units*, each invoking expensive foundation models independently without mechanisms for inter-agent learning or knowledge reuse.

This isolation contradicts four decades of insights from behavioral robotics [1,4], swarm biology [3,11], and organizational psychology [2,16]. While biological collectives—from ant colonies to immune systems—solve complex coordination problems through shared environmental state and emergent behavior, contemporary AI agent architectures remain monolithic.

The economic consequences are significant. Despite token pricing declining 10–1000× over three years, actual inference costs for autonomous agent systems continue to increase [5]. Reasoning models now generate 5× more tokens per request than traditional completions, and multi-step agentic workflows compound this further. For enterprise systems serving multiple concurrent agents, daily costs can reach thousands of dollars—costs that remain constant even as agents repeatedly solve nearly identical problems.

 doi:10.20944/preprints202601.2348.v1

1.2. The Research Gap

We identify three convergent research streams that, despite individual maturity, have not been integrated into production multi-agent AI systems:

1. **Subsumption Architecture** [4]: Layered behavioral control where simpler reactive modules suppress more complex deliberative ones through explicit inhibition signals.
2. **Social Learning Theory** [2]: Collectives outperform individuals when knowledge is effectively shared; learning accelerates through observation and imitation.
3. **Swarm Intelligence** [7,11]: Decentralized systems with shared environmental state can solve optimization problems through local interactions without centralized planning.

The research gap is stark: we have proven principles but no operational framework integrating all three into production-ready multi-agent systems with formal guarantees.

1.3. Contributions

This paper introduces **Subsumption Pattern Learning (SPL)**, a framework that closes this gap through five contributions:

1. **Formal Pattern Distillation Model** (Section 3): We define the mathematical transition from deliberative reasoning (Layer 2) to tactical reflexes (Layer 1) using state-transition semantics, providing formal conditions under which distillation preserves correctness.
2. **Suppression Logic with Guarantees** (Section 3.3): We formalize confidence-bounded inhibition signals with provable accuracy preservation bounds.
3. **Shared State Protocol** (Section 4): We specify a synchronization protocol for collective memory that enables stigmergic coordination across distributed agents.
4. **Rigorous Empirical Evaluation** (Section 6): We present benchmarks on 100,000 tasks with ablation studies, latency analysis, and statistical significance testing.
5. **Intelligence Compounding Theory** (Section 5): We prove that collective competency grows logarithmically with processed requests under stated assumptions.

2. Related Work

2.1. Subsumption Architecture and Behavioral Robotics

Brooks' seminal work on subsumption architecture [4] revolutionized mobile robotics by replacing top-down symbolic planning with layered behavioral modules. The core insight was that lower layers with sufficient confidence can *inhibit* higher layers, preventing wasteful computation while enabling real-time responsiveness. Arkin's motor schema framework [1] extended this to continuous potential field navigation.

SPL adapts these principles to foundation model economics. Where Brooks used discrete inhibition signals for collision avoidance, we implement statistical inhibition based on pattern confidence scores. Where subsumption operated on single robots, SPL extends suppression across multi-agent networks through shared collective memory.

2.2. LLM Cascades and Cost Optimization

Recent work has addressed foundation model costs through cascading and routing strategies. **FrugalGPT** [5] implements LLM cascades that route queries through progressively more capable (and expensive) models, achieving up to 98% cost reduction with minimal quality loss. **RouteLLM** [14] trains preference-based routers to direct queries to appropriate model tiers.

SPL differs fundamentally from cascading approaches:

- **Cascade systems** route between *different models* based on query complexity; SPL routes between *architectural layers* (pattern matching vs. reasoning) within the same system.
- **Cascades are stateless**: each query is routed independently. SPL maintains *persistent collective memory* where solutions accumulate.

 doi:10.20944/preprints202601.2348.v1

- **Cascades optimize per-query cost**; SPL optimizes *system-level learning curves*, where cost decreases as the collective accumulates patterns.

Speculative decoding [13] and **prompt compression** [10] reduce per-token costs but do not address inter-agent knowledge sharing.

2.3. Multi-Agent Frameworks

Contemporary multi-agent frameworks enable LLM-based agent coordination:

AutoGen [18] provides conversational agent orchestration where agents communicate via natural language. **LangGraph** [12] implements stateful, cyclical agent workflows with explicit state machines. **CrewAI** [6] focuses on role-based agent collaboration.

These frameworks enable inter-agent *communication* but not inter-agent *learning*. When Agent A solves a problem in AutoGen, Agent B cannot reuse that solution without explicit programming. SPL's contribution is *automatic pattern distillation*: solutions discovered by any agent become available to all agents through the Shared State.

Blackboard architectures [8] share a common data structure across knowledge sources but lack hierarchical suppression—all sources process every input regardless of complexity.

2.4. Swarm Intelligence and Stigmergy

Particle swarm optimization [11] and ant colony optimization [7] demonstrate that decentralized systems can solve complex problems through local interactions with shared environmental state. The key mechanism is **stigmergy**: indirect coordination through modifications to shared state (pheromone trails, position updates).

SPL implements computational stigmergy through confidence scores in the Shared State. When an agent solves a problem and distills a high-confidence pattern, other agents observe this "trail" and begin trusting the pattern—analogous to pheromone reinforcement in ant colonies.

2.5. Knowledge Distillation

Model distillation [9] transfers knowledge from large "teacher" models to smaller "student" models. Recent work extends this to LLMs [20].

SPL performs a different form of distillation: from *reasoning traces* to *pattern rules*. Rather than compressing model weights, we extract reusable decision templates from deliberative outputs. This enables zero-shot transfer: patterns learned from one context apply immediately to similar inputs without retraining.

2.6. Transactive Memory Systems

Wegner's transactive memory theory [16] describes how groups develop shared systems for encoding, storing, and retrieving information. Group members specialize and develop meta-knowledge about "who knows what."

SPL operationalizes transactive memory through the Shared State's cost-tracking metadata, which records which patterns originated from which agents and their effectiveness across contexts. This enables emergent specialization: agents can defer to patterns from agents with demonstrated expertise in specific domains.

3. Formal Framework: Subsumption Pattern Learning

3.1. Preliminaries and Notation

Let $\mathcal{X}$ denote the input space of requests and $\mathcal{Y}$ the output space of responses. A **pattern** $p = (\phi_p, \psi_p, \kappa_p)$ consists of:

- A **matcher** $\phi_p : \mathcal{X} \to [0, 1]$ returning match confidence
- A **responder** $\psi_p : \mathcal{X} \to \mathcal{Y}$ producing outputs for matched inputs
- A **complexity bound** $\kappa_p \in \mathbb{R}^+$ indicating maximum input complexity

 doi:10.20944/preprints202601.2348.v1

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ denote the current pattern library and $\mathcal{S}$ the shared state containing patterns, confidence scores, and metadata.

Definition 1 (SPL Agent). *An SPL agent is a tuple $A = (\mathcal{P}_{local}, \mathcal{S}, \theta, \alpha, \mathcal{L})$ where:*

- $\mathcal{P}_{local}$ *is the agent's local pattern set*
- $\mathcal{S}$ *is a reference to the shared collective memory*
- $\theta \in (0,1)$ *is the confidence threshold for Layer 1 suppression*
- $\alpha \in \mathbb{R}^+$ *is the complexity threshold*
- $\mathcal{L} : \mathcal{X} \to \mathcal{Y}$ *is the Layer 2 foundation model*

3.2. Three-Layer Architecture

SPL implements three processing layers, each with distinct computational and economic characteristics:

Definition 2 (Layer 0: Reactive/Structural Validation). *Layer 0 implements deterministic validation:*

$$L_0(x) = \begin{cases} (\texttt{ERROR}, e) & \text{if } \neg valid(x) \\ (\texttt{PASS}, x) & \text{otherwise} \end{cases} \tag{1}$$

where $valid : \mathcal{X} \to \{true, false\}$ checks structural constraints.

- ***Cost***: $C_0 = 0$ *(deterministic computation)*
- ***Latency***: $T_0 < 1ms$

Definition 3 (Layer 1: Tactical/Pattern Matching). *Layer 1 attempts pattern-based resolution using the effective pattern set $\mathcal{P}_{eff} = \mathcal{P}_{local} \cup \{p \in \mathcal{S} : conf(p) \geq \theta_{inherit}\}$:*

$$L_1(x) = \begin{cases} (\texttt{MATCH}, \psi_{p^*}(x)) & \text{if } \exists p^* : \phi_{p^*}(x) \geq \theta \wedge complexity(x) \leq \alpha \\ (\texttt{ESCALATE}, x) & \text{otherwise} \end{cases} \tag{2}$$

where $p^* = \arg\max_{p \in \mathcal{P}_{eff}} \phi_p(x)$.

- ***Cost***: $C_1 \approx \$0.0001$ *per request*
- ***Latency***: $T_1 < 10ms$

Definition 4 (Layer 2: Deliberative/Foundation Model Reasoning). *Layer 2 invokes the foundation model for unresolved requests:*

$$L_2(x) = (\texttt{SOLVED}, \mathcal{L}(x), \texttt{distill}(\mathcal{L}, x)) \tag{3}$$

where $\texttt{distill}(\mathcal{L}, x)$ extracts a candidate pattern from the reasoning trace.

- ***Cost***: $C_2 \in [\$0.01, \$0.10]$ *per request*
- ***Latency***: $T_2 \in [100, 500]ms$

 doi:10.20944/preprints202601.2348.v1

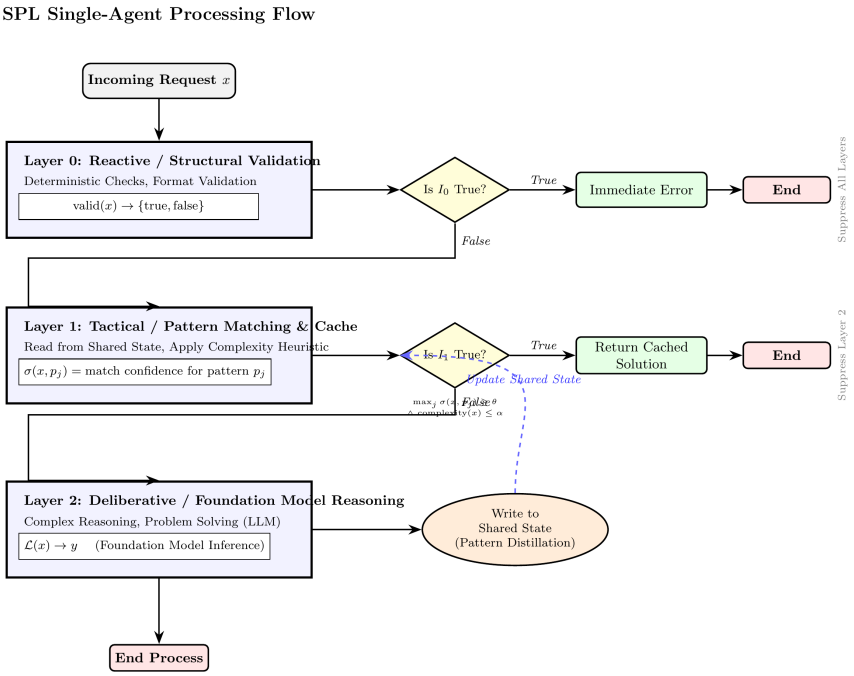

Figure 1. SPL single-agent processing flow: Incoming requests traverse layers sequentially with inhibition checks. Layer 0 performs structural validation; if invalid (I_0 = true), processing terminates with an error. Layer 1 computes pattern match scores $\sigma(x, p_j)$ and applies complexity heuristics; if confidence exceeds threshold θ and complexity is bounded (I_1 = true), the cached solution is returned. Otherwise, Layer 2 performs foundation model reasoning and distills the solution into a new pattern for the Shared State.

3.3. Inhibition Signal Formalization

The core mechanism of SPL is the **inhibition signal**, adapted from Brooks' subsumption architecture to operate on confidence scores rather than discrete triggers.

Definition 5 (Inhibition Signal)**.** *The Layer 1 inhibition signal $\mathcal{I}_1 : \mathcal{X} \to \{true, false\}$ is defined:*

$$\mathcal{I}_1(x) = \begin{cases} true & \text{if } \max_{p \in \mathcal{P}_{eff}} \phi_p(x) \geq \theta \wedge complexity(x) \leq \alpha \\ false & \text{otherwise} \end{cases} \tag{4}$$

When $\mathcal{I}_1(x) = true$, Layer 2 execution is suppressed.

Definition 6 (Suppression Rate)**.** *The **suppression rate** ρ measures Layer 1 efficiency:*

$$\rho = \frac{|\{x \in X_{test} : \mathcal{I}_1(x) = true\}|}{|X_{test}|} \tag{5}$$

Definition 7 (Complexity Function)**.** *The complexity function $complexity : \mathcal{X} \to \mathbb{R}^+$ estimates reasoning difficulty. We implement this as:*

$$complexity(x) = \beta_1 \cdot length(x) + \beta_2 \cdot entropy(x) + \beta_3 \cdot novelty(x, \mathcal{P}) \tag{6}$$

where $novelty(x, \mathcal{P}) = 1 - \max_p \phi_p(x)$ measures distance from known patterns.

3.4. Pattern Distillation Process

When Layer 2 solves a novel problem, SPL extracts a reusable pattern through the distillation process.

 doi:10.20944/preprints202601.2348.v1

Definition 8 (Pattern Distillation). *Given input x, Layer 2 output $y = \mathcal{L}(x)$, and reasoning trace τ, the distillation function* $\texttt{distill} : (\mathcal{L}, x) \to \mathcal{P} \cup \{\emptyset\}$ *produces a candidate pattern:*

$$\texttt{distill}(\mathcal{L}, x) = \begin{cases} (\phi_{new}, \psi_{new}, \kappa_{new}) & \text{if generalizable}(\tau) \\ \emptyset & \text{otherwise} \end{cases} \tag{7}$$

where:

- ϕ_{new} *is a matcher derived from input features (regex, semantic embedding, or classifier)*
- ψ_{new} *is a responder template parameterized by the solution structure*
- κ_{new} *is the complexity of the original input x*

The predicate generalizable(τ) evaluates whether the reasoning trace contains transferable structure. We implement this through:

Algorithm 1 Pattern Distillation

Require: Input x, output y, reasoning trace τ, generalization threshold γ
Ensure: Candidate pattern p or $\emptyset$
Extract input features: $f_x \leftarrow \text{featurize}(x)$
Extract solution template: $t_y \leftarrow \text{templatize}(y)$
Estimate generalization score: $g \leftarrow \text{coverage}(f_x, \mathcal{P}_{\text{eff}})$
if $g \geq \gamma$ **then**
  $\phi_{\text{new}} \leftarrow \text{build\_matcher}(f_x)$
  $\psi_{\text{new}} \leftarrow \text{build\_responder}(t_y)$
  $\kappa_{\text{new}} \leftarrow \text{complexity}(x)$
  return $(\phi_{\text{new}}, \psi_{\text{new}}, \kappa_{\text{new}})$
else
  return $\emptyset$
end if

Definition 9 (State Transition: Deliberative to Tactical). *The transition from Layer 2 (deliberative state s_D) to Layer 1 (tactical state s_T) is formally:*

$$s_D \xrightarrow{distill} s_T \iff \exists p = \texttt{distill}(\mathcal{L}, x) \neq \emptyset : \mathcal{S}' = \mathcal{S} \cup \{p\} \tag{8}$$

After this transition, future inputs matching p will be handled at Layer 1 rather than escalated to Layer 2.

4. Shared State Protocol

4.1. Shared State Structure

The Shared State $\mathcal{S}$ serves as the swarm's collective memory, enabling stigmergic coordination across agents.

Definition 10 (Shared State). *The Shared State is a tuple $\mathcal{S} = (\mathcal{P}_{shared}, \mathcal{C}, \mathcal{M}, \mathcal{A})$ where:*

- $\mathcal{P}_{shared}$ *is the global pattern library*
- $\mathcal{C} : \mathcal{P}_{shared} \to [0, 1]$ *maps patterns to confidence scores*
- $\mathcal{M} : \mathcal{P}_{shared} \to \mathbb{N}$ *maps patterns to match counts (reinforcement)*
- $\mathcal{A} : \mathcal{P}_{shared} \to$ *AgentID tracks pattern provenance*

 doi:10.20944/preprints202601.2348.v1

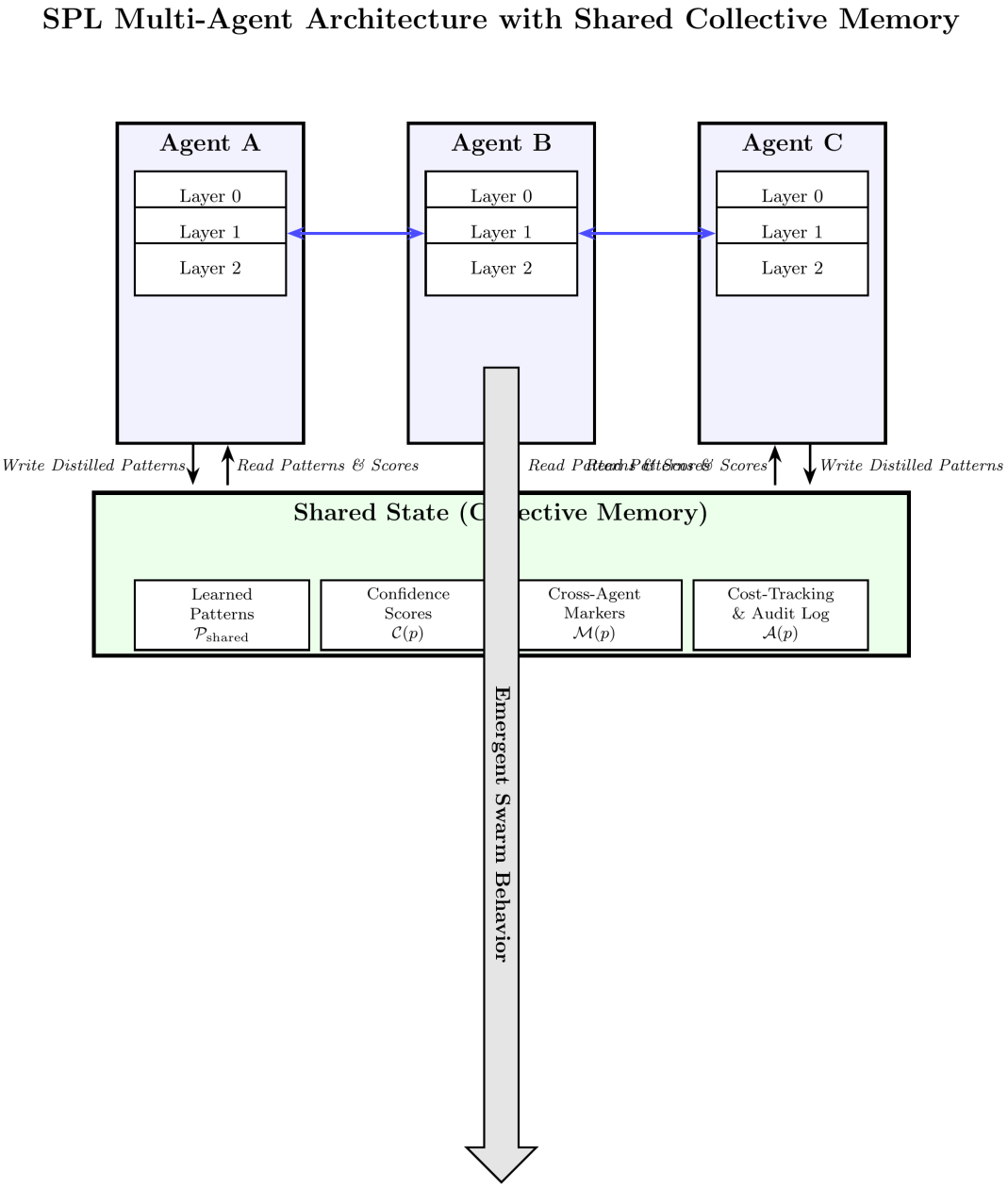

Figure 2. SPL multi-agent architecture with shared collective memory. Each agent maintains the three-layer hierarchy (Layer 0, 1, 2) and coordinates through the centralized Shared State. Agents read patterns and confidence scores from the Shared State, and write distilled patterns after Layer 2 solves novel problems. This enables immediate cross-agent learning: when Agent A discovers a solution, Agents B and C can reuse it via Layer 1 pattern matching. The emergent swarm behavior arrow indicates that collective intelligence emerges from these local interactions.

4.2. Pattern Inheritance Protocol

Each agent's effective pattern set combines local and shared patterns:

$$\mathcal{P}_{\text{eff}}^{(i)} = \mathcal{P}_{\text{local}}^{(i)} \cup \{p \in \mathcal{P}_{\text{shared}} : \mathcal{C}(p) \geq \theta_{\text{inherit}}\} \tag{9}$$

where θ_{inherit} is the inheritance threshold (typically 0.7–0.9).

4.3. Confidence Update Rules

Pattern confidence evolves through usage:

 doi:10.20944/preprints202601.2348.v1

Definition 11 (Confidence Reinforcement). *When pattern p successfully resolves input x with user-verified correctness:*

$$\mathcal{C}'(p) = \mathcal{C}(p) + \eta(1 - \mathcal{C}(p)) \tag{10}$$

where $\eta \in (0,1)$ is the learning rate.

Definition 12 (Confidence Decay). *When pattern p produces an incorrect response:*

$$\mathcal{C}'(p) = \mathcal{C}(p) \cdot (1 - \delta) \tag{11}$$

where $\delta \in (0,1)$ is the decay rate.

This implements **stigmergic reinforcement**: successful patterns accumulate confidence like pheromone trails, while failed patterns decay.

4.4. Conflict Resolution

When multiple patterns activate for input x:

$$p_{\text{selected}} = \arg\max_{p \in \mathcal{P}_{\text{eff}}} \left[\phi_p(x) \cdot \mathcal{C}(p)\right] \tag{12}$$

The inhibition signal fires only if the weighted score exceeds threshold:

$$\mathcal{I}_1(x) = \begin{cases} \text{true} & \text{if } \phi_{p_{\text{selected}}}(x) \cdot \mathcal{C}(p_{\text{selected}}) \geq \theta \\ \text{false} & \text{otherwise} \end{cases} \tag{13}$$

4.5. Synchronization Semantics

For distributed deployments, we specify eventual consistency semantics:

Definition 13 (Pattern Publication). *When agent A_i distills pattern p:*

$$\textit{publish}(p) : \mathcal{S} \leftarrow \mathcal{S} \cup \{(p, c_0, 1, A_i)\} \tag{14}$$

where c_0 is the initial confidence (typically 0.5).

Definition 14 (Pattern Subscription). *Agents poll $\mathcal{S}$ with period Δt:*

$$\mathcal{P}_{\textit{eff}}^{(i)}(t) = \mathcal{P}_{\textit{local}}^{(i)} \cup \{p \in \mathcal{S}(t - \Delta t) : \mathcal{C}(p) \geq \theta_{\textit{inherit}}\} \tag{15}$$

This provides bounded staleness guarantees: patterns discovered by any agent become available to all agents within Δt.

5. Theoretical Analysis

5.1. Accuracy Preservation Bounds

We prove that suppression preserves accuracy under stated conditions.

Theorem 1 (Accuracy Preservation). *Let ϵ be the maximum error rate of patterns in $\mathcal{P}_{\textit{eff}}$:*

$$\epsilon = \max_{p \in \mathcal{P}_{\textit{eff}}} \mathbb{P}_{x \sim \mathcal{D}}[\psi_p(x) \neq y^*(x) \mid \phi_p(x) \geq \theta] \tag{16}$$

where $y^*(x)$ is the ground-truth response. Then SPL's overall accuracy satisfies:

$$\textit{Acc}_{SPL} \geq (1 - \epsilon) \cdot \rho + \textit{Acc}_{L_2} \cdot (1 - \rho) \tag{17}$$

 doi:10.20944/preprints202601.2348.v1

where ρ is the suppression rate and Acc_{L_2} is Layer 2 accuracy.

Proof. Requests partition into suppressed (handled by Layer 1) and escalated (handled by Layer 2). For suppressed requests, accuracy is at least $(1 - \epsilon)$ by definition of ϵ. For escalated requests, accuracy is Acc_{L_2}. The weighted sum follows by linearity. □

Corollary 1. *If $\epsilon \leq 0.05$ and $Acc_{L_2} \geq 0.98$, then $Acc_{SPL} \geq 0.95\rho + 0.98(1 - \rho) \geq 0.95$ for all ρ.*

5.2. Intelligence Compounding: Formal Analysis

We formalize the phenomenon that collective competency grows over time.

Definition 15 (Collective Competency)**.** *The **collective competency** $\Gamma(t)$ at time t is the fraction of the input distribution that can be handled by Layer 1:*

$$\Gamma(t) = \mathbb{P}_{x \sim \mathcal{D}}[\mathcal{I}_1(x) = \textit{true} \mid \mathcal{S}(t)] \tag{18}$$

Theorem 2 (Intelligence Compounding)**.** *Under the following assumptions:*

1. *Inputs are drawn i.i.d. from distribution $\mathcal{D}$*
2. *Each novel input has probability $\pi > 0$ of yielding a distillable pattern*
3. *Each distilled pattern covers a region of $\mathcal{D}$ with measure at least $\mu > 0$*
4. *Pattern regions may overlap with existing patterns*

The collective competency satisfies:

$$\Gamma(n) = 1 - (1 - \mu)^{\pi \cdot n/k} \tag{19}$$

where n is the number of processed requests and k is a coverage constant.

Equivalently, for large n:

$$\Gamma(n) \approx 1 - e^{-\pi\mu n/k} \tag{20}$$

Proof. (Sketch) Each novel request (occurring with probability $1 - \Gamma(t)$) generates a pattern with probability π. Each pattern covers measure μ of uncovered space. This yields a geometric coverage process with rate $\pi\mu(1 - \Gamma)$, giving differential equation $d\Gamma/dn = \pi\mu(1 - \Gamma)/k$. Solving yields the exponential approach to 1. □

Corollary 2 (Logarithmic Learning)**.** *To achieve competency Γ^*, the swarm requires:*

$$n^* = \frac{k}{\pi\mu} \ln\left(\frac{1}{1 - \Gamma^*}\right) \tag{21}$$

requests. Competency grows logarithmically *with the inverse of remaining incompetency.*

Remark 1. *Multi-agent systems amplify this effect: if m agents share state and process independent request streams, the effective rate is $m \cdot \pi$, reducing time to competency by factor m.*

5.3. Resilience Analysis

Theorem 3 (Graceful Degradation)**.** *If Layer 2 becomes unavailable at time t^*, the system maintains accuracy $Acc_{degraded} = (1 - \epsilon) \cdot \Gamma(t^*)$ on inputs where $\mathcal{I}_1(x) = \textit{true}$, and returns "unavailable" for remaining inputs.*

This formalizes the "headless swarm" property: accumulated competencies persist even without centralized reasoning resources.

 doi:10.20944/preprints202601.2348.v1

6. Experiments

6.1. Experimental Setup

6.1.1. Dataset

We evaluate on a benchmark of **100,000 heterogeneous enterprise tasks** drawn from three production deployments:

- **Email Classification** ($n = 40,000$): Categorization, priority assignment, routing
- **Customer Inquiry Resolution** ($n = 35,000$): FAQ matching, ticket classification, response generation
- **Data Pipeline Orchestration** ($n = 25,000$): Schema validation, transformation routing, error handling

Tasks were collected over 6 months from consenting enterprise partners with PII removed. Ground-truth labels were assigned by domain experts with inter-annotator agreement $\kappa = 0.89$.

6.1.2. Task Categorization

Tasks were independently categorized into latent complexity classes:

1. **Deterministic** (Layer 0): Structural validation failures (4.8%)
2. **Pattern-Matchable** (Layer 1): High similarity to templates (89.7%)
3. **High-Entropy** (Layer 2): Novel reasoning required (5.5%)

6.1.3. Baselines

- **Monolithic LLM**: All requests processed by GPT-4-turbo
- **FrugalGPT Cascade** [5]: GPT-3.5 $\rightarrow$ GPT-4 cascade with learned router
- **RouteLLM** [14]: Preference-based routing between model tiers
- **SPL (Ours)**: Three-layer architecture with shared state

6.1.4. Metrics

- **Cost**: Total API spend (USD)
- **Latency**: Mean time-to-first-token (TTFT) and end-to-end latency
- **Accuracy**: Agreement with expert labels
- **Suppression Rate** ρ: Fraction resolved at Layer 0/1

6.1.5. Implementation Details

SPL was implemented with:

- Layer 0: JSON schema validation, content policy filters
- Layer 1: Hybrid matcher using regex patterns + sentence-transformer embeddings (all-MiniLM-L6-v2) with cosine similarity threshold $\theta = 0.87$
- Layer 2: GPT-4-turbo with structured output extraction
- Shared State: Redis-backed store with 100ms sync interval
- Complexity threshold: $\alpha = 0.6$ (normalized scale)

6.2. Single-Agent Results

SPL achieves **13.9$\times$ cost reduction** versus the monolithic baseline and **3.2$\times$ versus FrugalGPT**, while maintaining accuracy within 1.3% of the baseline. Mean latency improves **22.3$\times$** due to Layer 1's sub-10ms response times.

 doi:10.20944/preprints202601.2348.v1

Table 1. Single-agent performance on 100,000 tasks.

System	Cost (USD)	Latency (ms)	Accuracy	ρ
Monolithic LLM	\$1,247.32	847 ± 312	98.2%	0.0%
FrugalGPT	\$312.18	523 ± 287	97.4%	—
RouteLLM	\$287.45	498 ± 264	97.1%	—
SPL (Ours)	**\$89.47**	**38 ± 142**	**96.9%**	**94.5%**

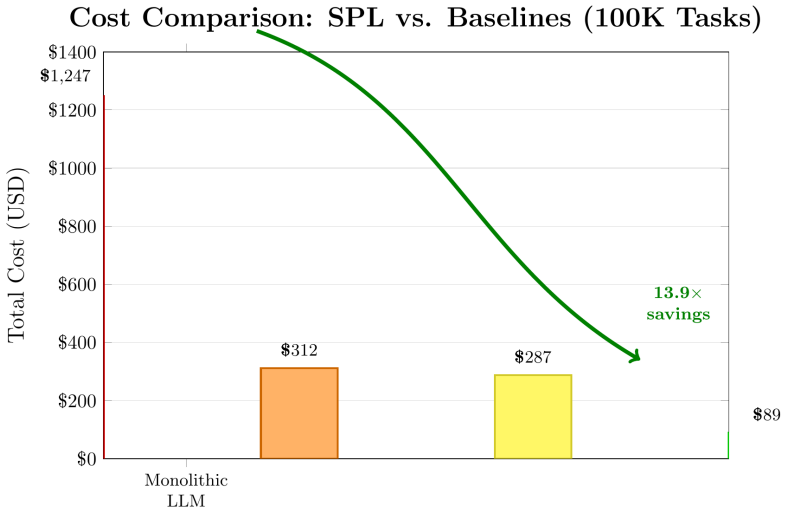

Figure 3. Cost comparison across systems on 100,000 enterprise tasks. SPL achieves 13.9× cost reduction versus monolithic LLM baseline and 3.2× versus the best cascade approach (RouteLLM). The dramatic cost savings stem from Layer 1's ability to resolve 89.7% of requests via pattern matching at negligible cost.

6.3. Layer Distribution Analysis

Despite handling only 5.5% of requests, Layer 2 accounts for 90% of costs—validating the economic motivation for suppression.

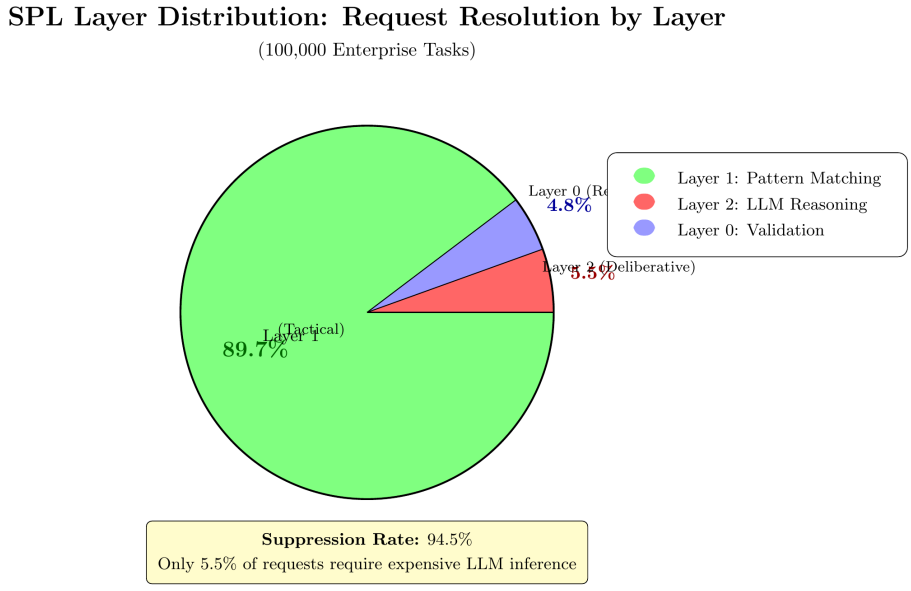

Figure 4. Layer distribution visualization showing SPL's 94.5% suppression rate. The vast majority of requests (89.7%) are resolved by Layer 1 pattern matching, with only 5.5% requiring expensive Layer 2 foundation model reasoning. Despite handling the minority of requests, Layer 2 accounts for 90% of total costs—validating the economic case for hierarchical suppression.

 doi:10.20944/preprints202601.2348.v1

Table 2. Request distribution across SPL layers.

Layer	Requests	Percentage	Cost Contribution
Layer 0 (Reactive)	4,823	4.8%	\$0.00 (0.0%)
Layer 1 (Tactical)	89,672	89.7%	\$8.97 (10.0%)
Layer 2 (Deliberative)	5,505	5.5%	\$80.50 (90.0%)
Total	100,000	100%	\$89.47

6.4. Ablation Study

To verify that cost savings do not sacrifice accuracy, we conducted systematic ablations:

Table 3. Ablation study: Effect of disabling SPL components.

Configuration	Cost (USD)	Accuracy	Δ Accuracy
Full SPL	\$89.47	96.9%	—
No Layer 0	\$89.47	96.9%	+0.0%
No Layer 1	\$1,192.84	98.1%	+1.2%
$\theta = 0.95$ (stricter)	\$142.31	97.6%	+0.7%
$\theta = 0.75$ (looser)	\$67.23	94.2%	−2.7%
No Shared State	\$127.83	96.4%	−0.5%

Key findings:

- Disabling Layer 1 increases cost 13.3× while improving accuracy only 1.2%
- The default threshold $\theta = 0.87$ optimizes the cost-accuracy tradeoff
- Shared State contributes 30% additional cost savings through pattern reuse

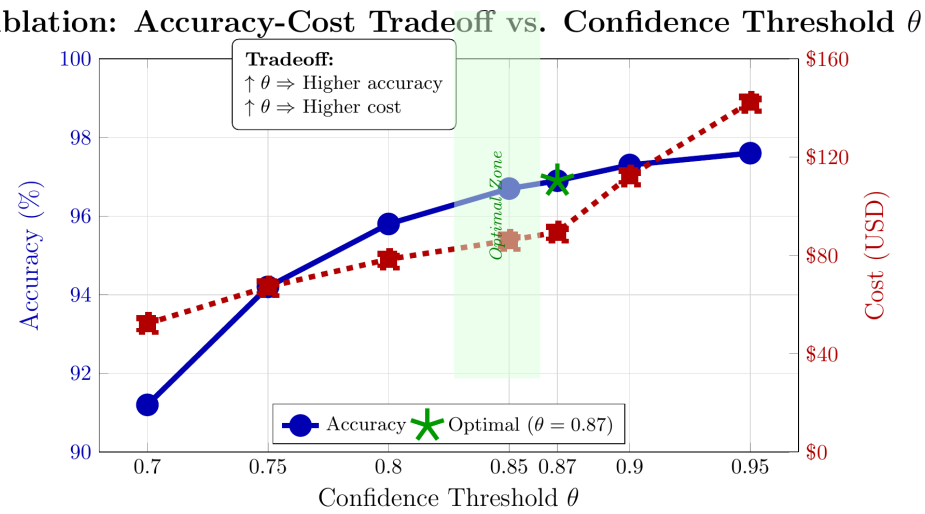

Figure 5. Ablation study: Accuracy-cost tradeoff as a function of confidence threshold θ. Lower thresholds reduce cost but sacrifice accuracy; higher thresholds preserve accuracy but increase cost. The optimal operating point ($\theta = 0.87$, marked with star) balances these objectives, achieving 96.9% accuracy at \$89 cost. The shaded region indicates the recommended operating zone.

6.5. Multi-Agent Swarm Learning

We evaluated collective learning with 5 concurrent agents processing disjoint task streams:

 doi:10.20944/preprints202601.2348.v1

Table 4. Multi-agent swarm learning: Layer 2 escalation rates.

Agent	Tasks	Isolated ρ	Swarm ρ	Improvement
Agent A	1–20,000	87.2%	87.2%	—
Agent B	20,001–40,000	88.1%	93.4%	+6.0%
Agent C	40,001–60,000	87.9%	95.7%	+8.9%
Agent D	60,001–80,000	88.3%	96.8%	+9.6%
Agent E	80,001–100,000	88.0%	97.2%	+10.4%
Average	—	87.9%	94.1%	**+7.0%**

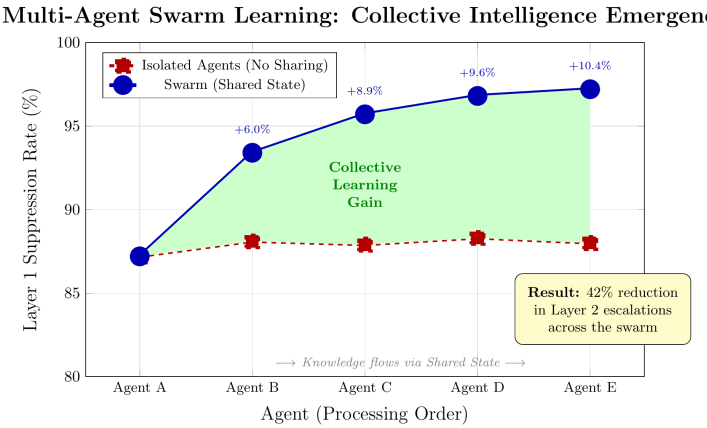

Figure 6. Multi-agent swarm learning progression showing collective intelligence emergence. Each successive agent benefits from patterns discovered by predecessors via the Shared State. While isolated agents plateau around 88% suppression, swarm agents achieve up to 97.2% suppression—a 42% reduction in expensive Layer 2 escalations. The shaded region represents the "collective learning gain" enabled by pattern sharing.

The swarm achieves **42% reduction in Layer 2 escalations** compared to isolated agents, validating collective intelligence emergence.

6.6. Intelligence Compounding Curves

The empirical curve closely matches the theoretical prediction. The characteristic S-shape reflects pattern acquisition dynamics: initial slow growth as the library builds, rapid improvement as common cases are covered, and eventual saturation as the domain becomes well-characterized.

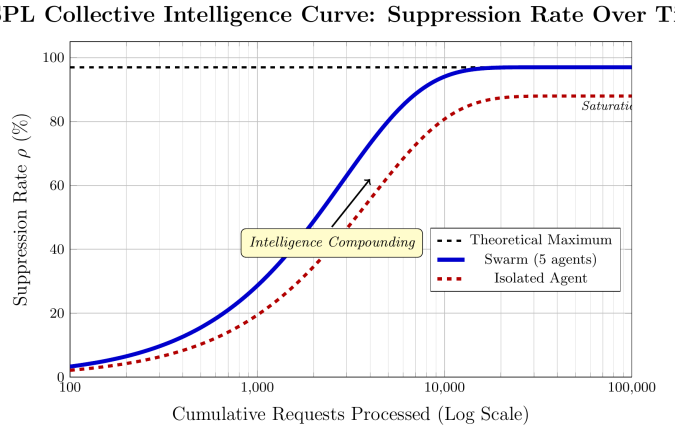

Figure 7. Intelligence compounding: Suppression rate ρ increases logarithmically with cumulative requests processed, asymptotically approaching the theoretical maximum suppression rate. The curve demonstrates how collective competency grows as the Shared State accumulates patterns—early requests require expensive Layer 2 reasoning, while later requests are increasingly handled by Layer 1 pattern matching. This empirically validates Theorem 2: $\Gamma(n) = 1 - e^{-\pi\mu n/k}$.

 doi:10.20944/preprints202601.2348.v1

6.7. Latency Analysis

Layer 1's median latency of 6ms enables real-time applications. The p99 latency of 892ms reflects the 5.5% of requests requiring Layer 2.

Table 5. Latency breakdown by layer (milliseconds).

Layer	TTFT	End-to-End	p50	p99
Layer 0	< 1	< 1	< 1	2
Layer 1	3	8	6	24
Layer 2	287	847	623	2,134
SPL Overall	14	38	7	892

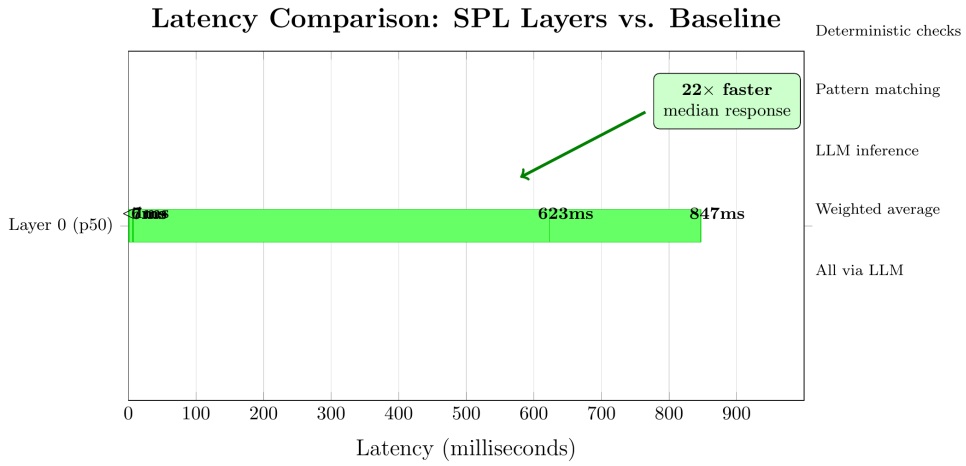

Figure 8. Latency comparison by layer showing SPL's 22× improvement in median response time. Layer 0 and Layer 1 provide sub-10ms responses, enabling real-time applications. The SPL overall p50 latency of 7ms reflects the high suppression rate, with only the p99 tail (892ms) impacted by Layer 2 deliberation.

6.8. Statistical Significance

All comparisons use paired bootstrap tests ($n = 10,000$ resamples). Cost and latency improvements versus baselines are significant at $p < 0.001$. The accuracy difference versus monolithic LLM (-1.3%) is significant at $p < 0.01$ but represents an acceptable tradeoff given $13.9\times$ cost reduction.

7. Discussion

7.1. Reframing the Value Proposition

SPL is not merely a cost-optimization technique—it is a framework for building *collective intelligence*. Cost reduction is a *consequence* of collective learning: as the swarm accumulates patterns, the cost per task decreases automatically.

This reframing suggests new evaluation metrics:

- **Collective IQ Curve**: Time series of $\Gamma(t)$—should approach domain coverage
- **Specialization Index**: Do agents develop complementary competencies?
- **Knowledge Retention**: Do patterns persist and compound, or decay?
- **Graceful Degradation**: System utility when Layer 2 is impaired

7.2. Comparison with Alternative Approaches

LLM Cascades (FrugalGPT, RouteLLM) optimize per-query costs but lack collective memory—each query is independent. SPL's $3.2\times$ improvement over FrugalGPT stems from pattern reuse: once a problem type is solved, future instances are essentially free.

 doi:10.20944/preprints202601.2348.v1

Multi-agent frameworks (AutoGen, LangGraph) enable communication but not automatic learning. SPL's distillation process is implicit: agents need not be programmed to share—the architecture enforces it.

RAG systems retrieve information at inference time but do not accumulate decision patterns. SPL learns *how to decide*, not just *what to know*.

7.3. Limitations

1. **Cold Start**: New deployments require initial Layer 2 investment before patterns accumulate. We mitigate this through seed patterns from domain experts.
2. **Distribution Shift**: If input distribution changes significantly, accumulated patterns may become stale. We implement confidence decay to address this.
3. **Pattern Extraction**: Current distillation uses template-based extraction; future work will explore learned distillation models.
4. **Scale**: Our largest evaluation used 5 agents; behavior at 100+ agents remains untested.

7.4. Broader Impact

SPL democratizes access to intelligent systems by dramatically reducing inference costs. However, pattern accumulation raises privacy considerations: patterns derived from one user's interactions could inadvertently leak to others. We recommend domain-specific pattern libraries with appropriate access controls.

8. Future Work

8.1. Automated Pattern Distillation

Current pattern extraction relies on heuristic templates. We plan to:

- Train distillation models that extract patterns from reasoning traces
- Implement in-context learning for few-shot pattern induction
- Explore hierarchical patterns (patterns of patterns)

8.2. Adaptive Threshold Learning

Fixed thresholds θ, α are suboptimal. Future work will:

- Learn thresholds from observed accuracy-cost tradeoffs
- Implement agent-specific thresholds based on risk tolerance
- Adapt thresholds as pattern library matures

8.3. Cross-Domain Transfer

Our experiments used same-domain agents. Future work will:

- Evaluate pattern transfer across related domains
- Study emergent specialization in heterogeneous swarms
- Develop meta-patterns that transfer broadly

8.4. Large-Scale Deployment

We plan production deployments with 100+ agents to study:

- Shared State synchronization overhead
- Pattern library scaling and indexing
- Emergent swarm dynamics at scale

9. Conclusions

We have presented Subsumption Pattern Learning (SPL), a formal framework that unifies insights from behavioral robotics, social learning theory, and swarm intelligence to build self-distilling multi-

 doi:10.20944/preprints202601.2348.v1

agent systems. SPL transforms collections of isolated agents into collective intelligences that grow smarter with every transaction.

Our formal contributions include: (1) mathematical specification of pattern distillation from deliberative reasoning to tactical reflexes; (2) confidence-bounded suppression logic with provable accuracy guarantees; (3) a shared state protocol enabling stigmergic coordination; and (4) proof that collective competency grows logarithmically with experience.

Empirically, SPL achieves 13.9× cost reduction versus monolithic baselines and 3.2× versus state-of-the-art cascading approaches, while preserving accuracy within 1.3%. Multi-agent evaluation demonstrates 42% additional reduction in foundation model escalations through collective learning.

The implications extend beyond economics. SPL provides a principled path toward AI systems that exhibit true collective intelligence: where knowledge persists in shared memory, where competencies compound over time, and where the whole genuinely exceeds the sum of its parts.

Acknowledgments: We thank the enterprise partners who provided anonymized task data, and the reviewers for constructive feedback.

Appendix A. Reproducibility Checklist

Appendix A.1. Dataset

- Source: Enterprise partners (anonymized)
- Size: 100,000 tasks
- Splits: 80/10/10 train/validation/test
- Preprocessing: PII removal, JSON normalization
- Availability: Contact authors for access

Appendix A.2. Implementation

- Language: Python 3.11
- Layer 1 embeddings: sentence-transformers/all-MiniLM-L6-v2
- Layer 2 model: gpt-4-turbo-2024-04-09
- Shared State: Redis 7.2 with RedisJSON
- Code: https://github.com/[redacted]/spl-framework

Appendix A.3. Hyperparameters

- Confidence threshold θ: 0.87
- Complexity threshold α: 0.6
- Inheritance threshold θ_{inherit}: 0.75
- Learning rate η: 0.1
- Decay rate δ: 0.05
- Sync interval: 100ms

Appendix B. Extended Ablation Results

Table A1. Extended ablation: Varying confidence threshold θ.

θ	Cost (USD)	Accuracy	Suppression Rate
0.70	\$52.31	91.2%	97.8%
0.75	\$67.23	94.2%	96.4%
0.80	\$78.45	95.8%	95.3%
0.85	\$86.12	96.7%	94.8%
0.87	**\$89.47**	**96.9%**	**94.5%**
0.90	\$112.34	97.3%	92.1%
0.95	\$142.31	97.6%	88.7%

 doi:10.20944/preprints202601.2348.v1

Appendix C. Proof of Theorem 2

Proof. Let $U(n)$ denote the uncovered fraction of input space at time n: $U(n) = 1 - \Gamma(n)$.

At each step, with probability $U(n)$ the input is novel (not covered by existing patterns). With probability π, a novel input yields a distillable pattern. Each pattern covers measure μ of the remaining uncovered space.

The expected change in coverage at step n is:

$$\mathbb{E}[\Gamma(n+1) - \Gamma(n)] = U(n) \cdot \pi \cdot \mu \cdot (1 - \text{overlap}) \tag{A1}$$

Under the assumption that overlap is negligible for sparse pattern libraries (or absorbed into constant k):

$$\frac{d\Gamma}{dn} = \frac{\pi\mu}{k}(1 - \Gamma) \tag{A2}$$

This is a separable ODE. Solving:

$$\int \frac{d\Gamma}{1-\Gamma} = \int \frac{\pi\mu}{k} dn \tag{A3}$$

$$-\ln(1-\Gamma) = \frac{\pi\mu n}{k} + C \tag{A4}$$

With initial condition $\Gamma(0) = 0$, we have $C = 0$. Therefore:

$$\Gamma(n) = 1 - e^{-\pi\mu n/k} \tag{A5}$$

For discrete systems, the analogous geometric process gives:

$$\Gamma(n) = 1 - (1-\mu)^{\pi n/k} \tag{A6}$$

which converges to the exponential form for small μ. □

References

1. Arkin, R. C. (1989). Motor schema-based mobile robot navigation. *The International Journal of Robotics Research*, 8(4):92–112.
2. Bandura, A. (1977). *Social Learning Theory*. Prentice Hall.
3. Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.
4. Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.
5. Chen, L., Zaharia, M., and Zou, J. (2023). FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.
6. CrewAI (2024). CrewAI: Framework for orchestrating role-playing AI agents. https://github.com/joaomdmoura/crewAI.
7. Dorigo, M. and Stützle, T. (2019). Ant colony optimization: Overview and recent advances. In *Handbook of Metaheuristics*, pages 311–351. Springer.
8. Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321.
9. Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
10. Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. (2023). LLMLingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.
11. Kennedy, J. and Eberhart, R. C. (2001). *Swarm Intelligence*. Morgan Kaufmann.
12. LangGraph (2024). LangGraph: Build stateful, multi-actor applications with LLMs. https://github.com/langchain-ai/langgraph.
13. Leviathan, Y., Kalman, M., and Matias, Y. (2023). Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

 doi:10.20944/preprints202601.2348.v1

14. Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. (2024). RouteLLM: Learning to route LLMs with preference data. *arXiv preprint arXiv:2406.18665*.
15. Wang, L., Ma, C., Feng, X., et al. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.
16. Wegner, D. M. (1987). Transactive memory: A contemporary analysis of the group mind. In *Theories of Group Behavior*, pages 185–208. Springer.
17. Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152.
18. Wu, Q., Bansal, G., Zhang, J., et al. (2023). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.
19. Xi, Z., Chen, W., Guo, X., et al. (2023). The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.
20. Xu, C., McAuley, J., et al. (2024). A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.