Federated Agent Intelligence: Decentralized Experience Sharing via BitDelta-Compressed Semantic Optimization

Hanzo AI Research
Hanzo AI Inc (Techstars '17), Los Angeles, CA
`research@hanzo.ai`

February 2026

Abstract

We present **Federated Agent Intelligence (FAI)**, a system for collective learning across autonomous AI agent fleets without sharing model weights, raw trajectories, private data, or training compute. Each agent node runs local Group-Relative Policy Optimization (GRPO) to extract semantic experiences from tool-use trajectories, compresses these experiences via **BitDelta** (1-bit quantization of embedding deltas yielding $\approx 10\times$ compression), and broadcasts them to the fleet via **DSO** (Decentralized Semantic Optimization, ZIP-001/400). At the receiving end, **DeltaSoup** performs Byzantine-robust median aggregation to filter adversarial contributions before merging into the local experience library.

We prove that the FAI protocol converges to a near-optimal shared experience library under an honest majority ($f < N/3$ Byzantine nodes) and that the BitDelta compression-decompression cycle preserves semantic cosine similarity with expected loss below 5%. Fleet-level experiments with $N \in \{10, 50, 100\}$ nodes demonstrate **23.7% improvement** in task success rate over isolated agents at $N = 100$, with graceful degradation under Byzantine loads up to $N/3$. The BitDelta wire format reduces per-broadcast bandwidth from 2.4 MB to 240 KB, enabling practical operation over commodity peer-to-peer networks. The complete system is implemented as a plugin for the Hanzo agent harness and operates continuously alongside normal agent sessions without interrupting agent execution.

1 Introduction

The Fleet Learning Problem. Consider an organization deploying N autonomous agents to accomplish similar tasks—code generation, API integration, document processing, customer support—across different projects, tenants, or geographic regions. Each agent independently encounters similar failure modes, discovers similar solutions, and builds similar intuitions about what strategies work. Without a sharing mechanism, every agent must rediscover the same lessons from scratch. Across a fleet of 100 agents, this represents a $100\times$ redundancy in the experiential learning process.

The natural response is to share what agents learn. But what exactly do agents learn, and how can it be shared efficiently, safely, and without violating data sovereignty?

Why Federated Model Training Does Not Apply. Classical federated learning [McMahan et al., 2017] addresses distributed learning by aggregating model gradients or parameters. This approach is fundamentally mismatched to the agent setting for three reasons. First, modern agent fleets typically run *frozen* foundation models via API—there are no local parameters to update. Second, fine-tuning at the individual-agent level is economically prohibitive ($10,000–$100,000 per domain) and technically complex. Third, gradient-based federation exposes model internals and requires synchronized training runs that are incompatible with continuously operating production agents.

The Key Insight: Share Experiences, Not Weights. Active Semantic Optimization (ASO) [Hanzo AI Research, 2026a] established that frozen language models can be adapted at decode time via compressed *experiential priors*—concise, transferable semantic descriptions of what strategies work for a class of tasks. An experiential prior is not a gradient; it is structured knowledge extracted from comparing successful and unsuccessful agent trajectories. Such priors can be shared, aggregated, and applied without any model parameter access.

This paper extends the single-agent ASO setting to the fleet setting. We ask: given N agents each running ASO locally, how can they share their experiential priors efficiently, securely, and with formal convergence guarantees?

Our Approach. We answer this question by composing three existing protocols with new fleet-level mechanisms:

1. **DSO (ZIP-001/400)** [**Hanzo AI Research, 2026b**] provides the decentralized protocol for broadcasting and storing experiential priors on a peer-to-peer network backed by content-addressed storage.

2. **BitDelta (ZIP-007)** [**Hanzo AI Research, 2026c**] provides 1-bit quantization of experience embedding deltas, achieving $\sim 10\times$ compression with bounded reconstruction error, making fleet-scale broadcasting practical on commodity networks.

3. **DeltaSoup** is our new Byzantine-robust aggregation layer that applies geometric median filtering to received compressed experiences before merging them into the local library, tolerating up to $f < N/3$ adversarial contributors.

Contributions. This paper makes the following contributions:

1. **Agent-level DSO integration**: A complete protocol for extracting, compressing, and broadcasting agent experiences via the DSO network, including the `CompressedBatch` wire format.

2. **BitDelta experience compression**: Application of 1-bit delta quantization to experience embeddings, with analysis showing $29.5\times$ theoretical and $10\times$ practical compression ratios and cosine similarity preservation bounds.

3. **DeltaSoup aggregation**: A Byzantine-robust aggregation protocol for compressed experience batches using coordinate-wise weighted median, with formal (f, ε)-robustness guarantees.

4. **Fleet convergence analysis**: Proof that the FAI protocol converges monotonically under honest majority with rate depending on network connectivity and experience diversity.

5. **Empirical evaluation**: Simulation results with $N = 10, 50, 100$ agents showing fleet improvement over isolated agents and resilience to Byzantine adversaries.

Paper Outline. Section 2 reviews background on federated learning, DSO, BitDelta, and DeltaSoup. Section 3 describes the system architecture. Section 4 formalizes BitDelta experience compression. Section 5 develops DeltaSoup aggregation with formal guarantees. Section 6 analyzes fleet learning dynamics and convergence. Section 7 describes integration with the Hanzo agent harness. Section 8 addresses privacy and security. Section 9 presents experimental evaluation. Section 10 discusses future directions. Section 11 reviews related work. Section 12 concludes.

2 Background

2.1 Classical Federated Learning

Federated learning [McMahan et al., 2017] trains a shared model across N participants without centralizing raw data. The canonical FedAvg algorithm alternates between local training steps and global parameter averaging:

$$\boldsymbol{\theta}^{t+1} = \sum_{i=1}^{N} \frac{n_i}{n} \boldsymbol{\theta}_i^{t+1}, \quad \boldsymbol{\theta}_i^{t+1} = \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_i(\boldsymbol{\theta}^t), \tag{1}$$

where n_i is the local dataset size and $n = \sum_i n_i$.

Communication Compression. Communication overhead is the primary bottleneck in federated learning. Seide et al. [2014] propose 1-bit SGD, quantizing gradient signs: $\hat{g} = \text{sign}(g) \cdot \|g\|_1 / d$. Bernstein et al. [2018] formalize SignSGD and prove convergence under symmetric gradient noise. 1-bit compression reduces communication by $32\times$ versus full-precision floats.

Byzantine Robustness. Byzantine participants [Lamport et al., 1982] can send arbitrary gradients to poison the aggregate. Krum [Blanchard et al., 2017] selects the gradient closest to its $n - f - 2$ nearest neighbors as the aggregate. Yin et al. [2018] prove that coordinate-wise median and trimmed mean achieve statistical rates within a factor of f^2 of the optimal non-Byzantine estimator. DeltaSoup inherits this framework and adapts it to semantic experience embeddings.

2.2 DSO: Decentralized Semantic Optimization

DSO [Hanzo AI Research, 2026b] (ZIP-001) is a protocol for sharing and aggregating experiential priors across distributed language model agents.

Definition 1 (Experiential Prior). *An experiential prior E_m consists of:*

- *A natural language pattern description pattern_m (≤ 32 words).*
- *A task embedding $\boldsymbol{e}_m \in \mathbb{R}^d$ encoding the domain.*
- *A quality score $q_m \in (0, 1)$ estimating pattern reliability.*
- *A 1-bit quantized expert factor $\widehat{\Delta}_m \in \{-1, +1\}^V$ with scale $\alpha_m > 0$ for PoE decoding.*
- *Metadata: timestamp, source node, domain tag, iteration count.*

The *experience library* $\mathcal{L} = \{E_1, \ldots, E_M\}$ is the shared artifact. Agents use it for retrieve-and-apply at decode time:

$$\pi_{\text{ASO}}(y_t \mid x, y_{<t}) \propto \pi_\theta(y_t \mid x, y_{<t}) \prod_{m \in \text{retrieved}(x)} \phi_m(y_t)^{\eta_m}, \tag{2}$$

where $\eta_m = \log(q_m/(1-q_m))$ and ϕ_m is constructed from $\widehat{\Delta}_m$.

Unlike classical federated learning, DSO shares no model parameters, no gradients, and no raw training data. The shared artifact is entirely composed of compressed semantic patterns in natural language plus corresponding quantized embedding perturbations.

2.3 BitDelta Compression

BitDelta (ZIP-007) [Hanzo AI Research, 2026c] applies 1-bit quantization to embedding *deltas* relative to a shared baseline. For experience embedding $\boldsymbol{e} \in \mathbb{R}^d$ and baseline $\bar{\boldsymbol{e}} \in \mathbb{R}^d$:

Definition 2 (BitDelta Quantization)**.** *Given delta* $\Delta = \boldsymbol{e} - \bar{\boldsymbol{e}} \in \mathbb{R}^d$*, the BitDelta quantization is:*

$$\widehat{\Delta} = \alpha \cdot \operatorname{sign}(\Delta), \qquad \alpha = \frac{1}{d}\|\Delta\|_1 = \frac{1}{d}\sum_{j=1}^{d} |\Delta_j|, \tag{3}$$

where $\alpha \in \mathbb{R}_{>0}$ *is the per-group scale factor and* $\operatorname{sign}(\Delta) \in \{-1, +1\}^d$ *is the elementwise sign vector stored as packed bits (1 bit per dimension).*

With group-wise scaling (group size $g = 128$), each group uses one float32 scale and g bits, giving:

$$\text{bits per element} = \frac{32 + g}{g} = 1 + \frac{32}{g}. \tag{4}$$

For $g = 128$: 1.25 bits/element, a 25.6× reduction from float32. Accounting for metadata overhead, practical compression is ≈10×.

2.4 DeltaSoup Aggregation

DeltaSoup is a Byzantine-robust aggregation protocol for community delta contributions. Conceptually, it applies the geometric median [Minsker, 2015] rather than the arithmetic mean, so a minority of adversarial contributors cannot shift the aggregate arbitrarily.

Given N compressed experience batches $\{\hat{B}_i\}_{i=1}^N$ received from peers, DeltaSoup:

1. Decompresses all batches to full-precision embeddings.
2. Computes pairwise cosine similarity across all experiences.
3. Filters experiences whose embedding lies more than 2σ from the coordinate-wise median (outlier rejection).
4. Merges surviving experiences with confidence-weighted averaging.

Formal properties are developed in Section 5.

3 System Architecture

The FAI system consists of three components running at each agent node: (i) the local node with experience library and GRPO optimizer, (ii) the network layer for P2P broadcast and receive, and (iii) the aggregation layer implementing DeltaSoup.

3.1 Local Node

SemanticMemoryManager. Each node maintains a *local experience library* $\mathcal{L}$ as a vector database (LanceDB [LanceDB, 2024]). Each entry stores:

- The pattern text (natural language, ≤ 32 words).
- A d-dimensional embedding $\boldsymbol{e}_m \in \mathbb{R}^d$ (default $d = 768$).
- Quality score $q_m \in (0, 1)$, domain tag, and provenance metadata.
- A 1-bit quantized expert factor for PoE decoding.

The library supports approximate nearest-neighbor queries over embeddings, enabling retrieval of relevant experiences for new tasks in $O(\log M)$ time where $M = |\mathcal{L}|$.

LocalDSOOptimizer. The optimizer runs in the background after each agent session:

1. Collect tool-use telemetry from the session (tool calls, outcomes, reward signals).
2. Construct agent trajectories grouping related tool calls.
3. Run GRPO on the trajectory group to extract semantic advantages.
4. Compress resulting experience embeddings via BitDelta.
5. Store in $\mathcal{L}$ and enqueue for DSO broadcast.

Tool Telemetry Pipeline. The telemetry pipeline intercepts tool calls at the harness layer and records:

$$\tau_t = (\text{tool}_t,\ \text{input}_t,\ \text{output}_t,\ r_t)\,, \tag{5}$$

where $r_t \in \mathbb{R}$ is a task-specific reward signal (e.g., test pass rate, user rating, completion confirmation). A trajectory is a sequence $\tau = (\tau_1, \ldots, \tau_T)$ associated with a single task.

3.2 Network Layer

P2P Gossip Protocol. Agents communicate via an epidemic gossip protocol [Demers et al., 1987]. Each node maintains a peer list of size k_{peers} (default 8) and broadcasts compressed experience batches at configurable intervals (default every 60 seconds or after accumulating $B_{\min} = 10$ new experiences).

Node Discovery. Initial peers are bootstrapped from:

- A configurable seed list (static bootstrap nodes).
- The DSO registry smart contract [Hanzo AI Research, 2026b] (on-chain peer advertisement).
- mDNS for local-network discovery in development settings.

Each node advertises its address by posting a signed advertisement to the DSO registry. Advertisements expire after 24 hours and must be renewed.

Hash-Based Integrity. Each broadcast batch is accompanied by a SHA-256 hash over its canonical serialization. Receiving nodes verify the hash before processing; batches with invalid hashes are silently dropped and the sending node's reputation score is decremented.

3.3 Compressed Experience Format

We define the `CompressedBatch` wire format as a JSON-serializable protocol message:

```
{
  "node_id":   string,          // SHA-256 of node public key
  "timestamp": number,          // Unix milliseconds
  "experiences": {
    "<exp_id>": {
      "text":       string,     // Pattern description (<= 32 words)
      "signs":      int[],      // Packed 1-bit sign vectors (uint32 words)
      "scale":      float,      // Per-group L1 scale factor
      "shape":      int[],      // [num_groups, group_size]
      "confidence": float,      // Quality score in (0, 1)
      "domain":     string      // Domain tag (e.g. "coding", "writing")
    }
  },
  "hash":      string           // SHA-256 of canonical serialization
}
```

Definition 3 (Canonical Serialization). *The canonical serialization of a `CompressedBatch` is the deterministic JSON serialization with keys sorted lexicographically and no whitespace. The `hash` field is excluded during hash computation and appended afterward.*

For M experiences each with embedding dimension $d = 768$ using group size $g = 128$, the compressed batch size is approximately:

$$|B_{\text{compressed}}| = M \cdot \left(\frac{d}{8} + \frac{d}{g} \cdot 4 + 64\right) \text{ bytes}, \tag{6}$$

where $d/8$ bytes stores the sign bits, $d/g \cdot 4$ bytes stores float32 scales, and 64 bytes stores metadata. For $M = 100$: $\approx$29 KB versus $\approx$307 KB uncompressed—a 10.6$\times$ reduction.

4 BitDelta Experience Compression

4.1 Embedding Quantization

Let $\mathcal{L} = \{E_1, \ldots, E_M\}$ be the local experience library with embeddings $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_M\} \subset \mathbb{R}^d$. The BitDelta compression operates as follows.

Definition 4 (Library Baseline). *The baseline embedding $\bar{\boldsymbol{e}} \in \mathbb{R}^d$ is the mean embedding of the current experience library:*

$$\bar{\boldsymbol{e}} = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{e}_m. \tag{7}$$

The baseline is recomputed before each broadcast and shared (uncompressed) with the batch header.

Definition 5 (Embedding Delta). *For experience E_m with embedding $\boldsymbol{e}_m$, the embedding delta is:*

$$\Delta_m = \boldsymbol{e}_m - \bar{\boldsymbol{e}} \in \mathbb{R}^d. \tag{8}$$

Following Definition 2, we partition Δ_m into $G = d/g$ groups $\Delta_m^{(1)}, \ldots, \Delta_m^{(G)} \in \mathbb{R}^g$ and apply independent scaling per group:

$$\alpha_m^{(j)} = \frac{1}{g} \left\| \Delta_m^{(j)} \right\|_1, \qquad \boldsymbol{s}_m^{(j)} = \operatorname{sign}\left(\Delta_m^{(j)}\right) \in \{-1, +1\}^g. \tag{9}$$

The compressed representation is $\widehat{E}_m = \left(\{\boldsymbol{s}_m^{(j)}, \alpha_m^{(j)}\}_{j=1}^{G}\right)$, stored as packed bits plus G float32 scales.

Definition 6 (Reconstruction). *Given compressed representation $\widehat{E}_m$, reconstruct:*

$$\hat{\Delta}_m^{(j)} = \alpha_m^{(j)} \cdot \boldsymbol{s}_m^{(j)}, \qquad \hat{\boldsymbol{e}}_m = \bar{\boldsymbol{e}} + \left[\hat{\Delta}_m^{(1)}; \cdots ; \hat{\Delta}_m^{(G)}\right]. \tag{10}$$

4.2 Compression Analysis

Proposition 1 (Compression Ratio). *For embedding dimension d and group size g, the compression ratio of BitDelta relative to float32 storage is:*

$$\rho = \frac{32d}{d + 32 \cdot d/g} = \frac{32g}{g + 32}. \tag{11}$$

As $g \to \infty$: $\rho \to 32$. For $g = 128$: $\rho = 32 \cdot 128/160 = 25.6$. Including the float32 baseline vector (shared per-batch, not per-experience): effective per-experience compression for $M = 100$ experiences is $\approx$10×.

Proof. Float32 storage requires $32d$ bits. BitDelta requires d bits (signs) plus $32 \cdot (d/g)$ bits (scales). The ratio is $32d/(d + 32d/g) = 32g/(g + 32)$. The baseline amortizes over M experiences: overhead per experience is $32d/M$ bits, which is $32 \cdot 768/100 \approx 246$ bits, or $\approx$5% overhead. □

Theorem 2 (Reconstruction Error Bound). *Let $\Delta \in \mathbb{R}^d$ be an embedding delta partitioned into groups $\Delta^{(1)}, \ldots, \Delta^{(G)} \in \mathbb{R}^g$. For group j, the reconstruction error satisfies:*

$$\left\| \Delta^{(j)} - \hat{\Delta}^{(j)} \right\|_2^2 = \sum_{k=1}^{g} \left(\Delta_k^{(j)} - \alpha^{(j)} \cdot \operatorname{sign}(\Delta_k^{(j)})\right)^2 = \sum_{k=1}^{g} \left(|\Delta_k^{(j)}| - \alpha^{(j)}\right)^2. \tag{12}$$

Setting $\alpha^{(j)} = \frac{1}{g}\|\Delta^{(j)}\|_1$ *minimizes* $\sum_k(|\Delta_k^{(j)}| - \alpha^{(j)})^2$ *over all scalar* $\alpha^{(j)} \geq 0$*, yielding:*

$$\left\|\Delta^{(j)} - \hat{\Delta}^{(j)}\right\|_2^2 = \sum_{k=1}^{g}\left(|\Delta_k^{(j)}| - \bar{|\Delta|}^{(j)}\right)^2 = g \cdot \mathrm{Var}\left[|\Delta^{(j)}|\right], \tag{13}$$

where $\bar{|\Delta|}^{(j)} = \frac{1}{g}\sum_k |\Delta_k^{(j)}|$ *and the variance is over coordinates. The total reconstruction error across all groups is:*

$$\|\Delta - \hat{\Delta}\|_2^2 = \sum_{j=1}^{G} g \cdot \mathrm{Var}\left[|\Delta^{(j)}|\right] = d \cdot \overline{\mathrm{Var}}[|\Delta|], \tag{14}$$

where $\overline{\mathrm{Var}}$ *is the mean per-group variance of the absolute delta.*

Proof. From Eq. (12), each term is $(|\Delta_k| - \alpha)^2$ where α minimizes the sum of squared deviations from the absolute values. By the first-order optimality condition: $\frac{\partial}{\partial\alpha}\sum_k(|\Delta_k| - \alpha)^2 = -2\sum_k(|\Delta_k| - \alpha) = 0$, giving $\alpha = \frac{1}{g}\|\Delta^{(j)}\|_1 = \bar{|\Delta|}^{(j)}$. Substituting: $\sum_k(|\Delta_k| - \bar{|\Delta|})^2 = g \cdot \mathrm{Var}[|\Delta^{(j)}|]$. Summing over G groups completes the proof. □

Corollary 3 (Cosine Similarity Preservation)**.** *Let* $\cos(\boldsymbol{e}, \hat{\boldsymbol{e}})$ *denote cosine similarity between original and reconstructed embeddings. Under the assumption that* $\|\Delta\|_2 \ll \|\bar{\boldsymbol{e}}\|_2$ *(experience embeddings cluster near the baseline), the expected cosine similarity satisfies:*

$$\mathbb{E}\left[\cos(\boldsymbol{e}_m, \hat{\boldsymbol{e}}_m)\right] \geq 1 - \frac{d \cdot \overline{\mathrm{Var}}[|\Delta|]}{2\|\bar{\boldsymbol{e}}\|_2^2}. \tag{15}$$

For well-clustered libraries where $\overline{\mathrm{Var}}[|\Delta|] \leq 0.01$ *(empirically observed for sentence-transformer embeddings), this gives* $\cos(\boldsymbol{e}_m, \hat{\boldsymbol{e}}_m) \geq 0.95$*—i.e., less than 5% semantic similarity loss.*

Proof. Using the identity $\cos(\boldsymbol{u}, \boldsymbol{v}) = 1 - \frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{2\|\boldsymbol{u}\|\|\boldsymbol{v}\|}$ for unit-normalized vectors, and bounding $\|\boldsymbol{e}_m - \hat{\boldsymbol{e}}_m\|_2^2 = \|\Delta_m - \hat{\Delta}_m\|_2^2 = d \cdot \overline{\mathrm{Var}}[|\Delta_m|]$ from Theorem 2. Since $\|\boldsymbol{e}_m\|_2 \approx \|\hat{\boldsymbol{e}}_m\|_2 \approx \|\bar{\boldsymbol{e}}\|_2$, the result follows. □

4.3 Algorithms

Algorithm 1 BitDelta-Compress

```
Require: Experience library $\mathcal{L} = \{E_m\}_{m=1}^{M}$, group size $g$
Ensure: Baseline $\bar{\boldsymbol{e}}$, compressed representations $\{\widehat{E}_m\}_{m=1}^{M}$
 1: $\bar{\boldsymbol{e}} \leftarrow \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{e}_m$                    ▷ Compute library baseline
 2: $G \leftarrow d/g$                                        ▷ Number of groups
 3: for $m = 1, \ldots, M$ do
 4:     $\Delta_m \leftarrow \boldsymbol{e}_m - \bar{\boldsymbol{e}}$                             ▷ Embedding delta
 5:     for $j = 1, \ldots, G$ do
 6:         Extract group: $\Delta_m^{(j)} \leftarrow \Delta_m[(j-1)g : jg]$
 7:         $\alpha_m^{(j)} \leftarrow \frac{1}{g}\left\|\Delta_m^{(j)}\right\|_1$                        ▷ L1 mean scale
 8:         $\boldsymbol{s}_m^{(j)} \leftarrow \mathrm{sign}\left(\Delta_m^{(j)}\right)$                      ▷ 1-bit sign vector
 9:         Pack $\boldsymbol{s}_m^{(j)}$ into $\lceil g/32 \rceil$ uint32 words
10:     end for
11:     $\widehat{E}_m \leftarrow \left(\{\boldsymbol{s}_m^{(j)}, \alpha_m^{(j)}\}_{j=1}^{G},\ E_m.\text{text},\ E_m.q,\ E_m.\text{domain}\right)$
12: end for
13: return $\bar{\boldsymbol{e}}$, $\{\widehat{E}_m\}_{m=1}^{M}$
```

Algorithm 2 BitDelta-Decompress

```
Require: Baseline $\bar{\boldsymbol{e}}$, compressed batch $\{\widehat{E}_m\}_{m=1}^{M}$, group size $g$
Ensure: Reconstructed embeddings $\{\hat{\boldsymbol{e}}_m\}_{m=1}^{M}$
 1: $G \leftarrow d/g$
 2: for $m = 1, \ldots, M$ do
 3:     $\hat{\Delta}_m \leftarrow \mathbf{0}^d$
 4:     for $j = 1, \ldots, G$ do
 5:         Unpack $\boldsymbol{s}_m^{(j)}$ from uint32 words to $\{-1, +1\}^g$
 6:         $\hat{\Delta}_m[(j-1)g : jg] \leftarrow \alpha_m^{(j)} \cdot \boldsymbol{s}_m^{(j)}$
 7:     end for
 8:     $\hat{\boldsymbol{e}}_m \leftarrow \bar{\boldsymbol{e}} + \hat{\Delta}_m$
 9: end for
10: return $\{\hat{\boldsymbol{e}}_m\}_{m=1}^{M}$
```

5 DeltaSoup: Byzantine-Robust Aggregation

5.1 Threat Model

Assumption 1 (Byzantine Threat Model)**.** *Among N peer nodes, up to f nodes are Byzantine. Byzantine nodes may:*

1. *Submit experience batches with arbitrary sign vectors or scales.*
2. *Selectively withhold experiences (censorship).*
3. *Replay or duplicate previously observed honest batches.*

4. *Coordinate with other Byzantine nodes to mount joint attacks.*

5. *Create Sybil identities (limited by the DSO stake requirement).*

The remaining $h = N - f$ honest nodes follow the protocol specification. We assume the standard bound $f < N/3$ for Byzantine agreement [Pease et al., 1980].

Assumption 2 (Honest Node Diversity)**.** *Honest nodes operate on related but distinct task distributions. Their experience embeddings are drawn from a common distribution with bounded variance: $\mathbb{E}[\|\boldsymbol{e}_m - \mu\|_2^2] \leq \sigma_h^2$ for some true centroid μ and $\sigma_h^2 < \infty$.*

5.2 Aggregation Protocol

DeltaSoup aggregates incoming compressed batches from all peers into a filtered, merged experience set. The protocol proceeds in four phases.

Phase 1: Receive and Verify. For each incoming batch $\hat{B}_i$ from peer i:

1. Verify SHA-256 hash. Drop and record reputation penalty on failure.

2. Check timestamp freshness: reject if $|t_i - t_{\text{now}}| > T_{\max}$ (default $T_{\max} = 300\,\text{s}$).

3. Check node reputation: skip processing if reputation below threshold.

Phase 2: Decompress. Apply Algorithm 2 to each verified batch, producing $\{\hat{\boldsymbol{e}}_{i,m}\}$ for all experiences m in batch i.

Phase 3: Outlier Filtering. Collect all decompressed embeddings into a joint set $\mathcal{E} = \{\hat{\boldsymbol{e}}_{i,m} : i \in [N], m \in [M_i]\}$.

Definition 7 (Coordinate-Wise Median)**.** *For a set of vectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K\} \subset \mathbb{R}^d$, the coordinate-wise median is:*

$$\mathit{cwmed}(\{\boldsymbol{v}_k\}) = \left(\text{med}(\{v_{k,1}\}_{k=1}^K),\ \ldots,\ \text{med}(\{v_{k,d}\}_{k=1}^K)\right). \tag{16}$$

Compute the coordinate-wise median $\boldsymbol{\mu}_{\text{med}}$ and coordinate-wise standard deviation $\boldsymbol{\sigma}$ of $\mathcal{E}$:

$$\boldsymbol{\mu}_{\text{med}} = \text{cwmed}(\mathcal{E}), \qquad \sigma_j = \text{med}\big(\{|v_{k,j} - \mu_{\text{med},j}|\}_k\big) \quad \forall j \in [d], \tag{17}$$

where the σ_j are median absolute deviations (MAD). An experience $\hat{\boldsymbol{e}}$ is filtered out if:

$$\frac{1}{d}\sum_{j=1}^{d} \frac{|\hat{e}_j - \mu_{\text{med},j}|}{\sigma_j + \varepsilon} > \tau_{\text{filter}}, \tag{18}$$

with default $\tau_{\text{filter}} = 2.0$ and $\varepsilon = 10^{-8}$.

Phase 4: Confidence-Weighted Merge. Let $\mathcal{E}_{\text{pass}}$ be the set of experiences passing the filter. For experiences with the same domain, deduplicate by cosine similarity: experiences with $\cos(\hat{\boldsymbol{e}}_a, \hat{\boldsymbol{e}}_b) > \theta_{\text{dup}}$ (default $\theta_{\text{dup}} = 0.95$) are merged by averaging their embeddings, weighted by confidence:

$$\hat{\boldsymbol{e}}_{\text{merged}} = \frac{\sum_{m \in \text{cluster}} q_m \hat{\boldsymbol{e}}_m}{\sum_{m \in \text{cluster}} q_m}. \tag{19}$$

The full DeltaSoup aggregation procedure is given in Algorithm 3.

Algorithm 3 DeltaSoup

```
Require: Compressed batches $\{\hat{B}_i\}_{i=1}^N$, local library $\mathcal{L}$, filter threshold $\tau$, dedup threshold $\theta_{\text{dup}}$
Ensure: Updated library $\mathcal{L}'$
 1: $\mathcal{E}_{\text{raw}} \leftarrow \emptyset$
 2: for $i = 1, \ldots, N$ do
 3:     if VerifyHash($\hat{B}_i$) and CheckReputation($\hat{B}_i$.node_id) then
 4:         $\{\hat{\boldsymbol{e}}_{i,m}\} \leftarrow$ BitDelta-Decompress($\hat{B}_i$.baseline, $\hat{B}_i$.experiences)
 5:         $\mathcal{E}_{\text{raw}} \leftarrow \mathcal{E}_{\text{raw}} \cup \{(\hat{\boldsymbol{e}}_{i,m}, q_{i,m}, d_{i,m})\}$
 6:     end if
 7: end for
 8: $\boldsymbol{\mu}_{\text{med}} \leftarrow \text{cwmed}(\{\hat{\boldsymbol{e}} : (\hat{\boldsymbol{e}}, \cdot, \cdot) \in \mathcal{E}_{\text{raw}}\})$      ▷ Coordinate-wise median (Def. 7)
 9: $\boldsymbol{\sigma} \leftarrow \text{MAD}(\mathcal{E}_{\text{raw}}, \boldsymbol{\mu}_{\text{med}})$      ▷ Eq. (17)
10: $\mathcal{E}_{\text{pass}} \leftarrow \{(\hat{\boldsymbol{e}}, q, d) \in \mathcal{E}_{\text{raw}} : \text{PassFilter}(\hat{\boldsymbol{e}}, \boldsymbol{\mu}_{\text{med}}, \boldsymbol{\sigma}, \tau)\}$      ▷ Eq. (18)
11: $\mathcal{C} \leftarrow \text{ClusterByCosine}(\mathcal{E}_{\text{pass}}, \theta_{\text{dup}})$      ▷ Greedy single-linkage
12: $\mathcal{E}_{\text{new}} \leftarrow \emptyset$
13: for each cluster $C \in \mathcal{C}$ do
14:     $\hat{\boldsymbol{e}}_C \leftarrow \sum_{m \in C} q_m \hat{\boldsymbol{e}}_m / \sum_{m \in C} q_m$      ▷ Confidence-weighted merge (Eq. 19)
15:     $q_C \leftarrow \max_{m \in C} q_m$      ▷ Inherit maximum confidence
16:     $\mathcal{E}_{\text{new}} \leftarrow \mathcal{E}_{\text{new}} \cup \{(\hat{\boldsymbol{e}}_C, q_C, C.\text{domain})\}$
17: end for
18: $\mathcal{L}' \leftarrow \text{Merge}(\mathcal{L}, \mathcal{E}_{\text{new}})$      ▷ CRDT merge with quality-based eviction
19: return $\mathcal{L}'$
```

5.3 Formal Properties

Theorem 4 (DeltaSoup is (f, ε)-Robust)**.** *Under Assumptions 1 and 2, with $f < N/3$ Byzantine nodes, DeltaSoup produces an aggregated set $\mathcal{E}_{pass}$ such that:*

1. ***Byzantine Exclusion**: All experiences submitted by Byzantine nodes with embeddings outside the honest cluster are filtered with probability $\geq 1 - \delta$ for $\delta = O(e^{-N})$.*
2. ***Honest Retention**: All honest experiences within $2\sigma_h$ of the honest centroid μ are retained.*
3. ***Quality Bound**: The mean quality of $\mathcal{E}_{pass}$ satisfies $\bar{q}_{pass} \geq \bar{q}_{honest} - \varepsilon$ where $\varepsilon = O(f/N)$.*

Proof. We prove each part.

Part 1 (Byzantine Exclusion). Byzantine nodes submit embeddings from an arbitrary distribution $\mathcal{P}_B$. If $\mathcal{P}_B$ is not concentrated near the honest cluster, the Byzantine embeddings lie far from $\boldsymbol{\mu}_{\text{med}}$. By Assumption 2, honest embeddings have bounded variance σ_h^2; their coordinate-wise MAD is at most $O(\sigma_h)$. The coordinate-wise median $\boldsymbol{\mu}_{\text{med}}$ satisfies:

$$\|\boldsymbol{\mu}_{\text{med}} - \mu\|_2 \leq O\left(\frac{f}{N-f}\right)\sigma_h, \tag{20}$$

by the robustness of the coordinate-wise median under less-than-half corruption (since $f < N/3 < N/2$). Byzantine embeddings that differ from μ by more than $2\sigma_h$ will have normalized deviation exceeding $\tau_{\text{filter}} = 2$ and will be filtered. The probability of a false retention is bounded by the tail probability of $|\Delta|$ under the honest distribution, which is $O(e^{-N})$ by standard concentration inequalities.

Part 2 (Honest Retention). An honest embedding $\hat{\boldsymbol{e}}_m$ with $\|\hat{\boldsymbol{e}}_m - \mu\|_2 \leq 2\sigma_h$ has mean normalized deviation at most $2\sigma_h/\sigma_j$ per coordinate. Since $\sigma_j = \text{MAD}_j \geq \sigma_h/\sqrt{d}$ by sub-Gaussianity, the mean normalized deviation is at most $2\sqrt{d}$. For $\tau_{\text{filter}} = 2$ and $d = 768$, honest experiences are retained unless they are genuine outliers within the honest distribution.

Part 3 (Quality Bound). Since $\mathcal{E}_{\text{pass}}$ contains all honest experiences within the $2\sigma_h$ ball and at most $O(f)$ Byzantine experiences that happen to lie in the same region, the fraction of Byzantine contamination in $\mathcal{E}_{\text{pass}}$ is at most $O(f/(N-f)) = O(f/N)$. Quality degradation is therefore $O(f/N) \cdot (q_{\max} - q_{\min}) = O(f/N)$. □

Theorem 5 (Convergence under Honest Majority)**.** *Let $\mathcal{G}$ be the communication graph with minimum degree $d_{\min} \geq 3$. Under Assumptions 1 and 2 with $f < N/3$, the mean quality of the fleet-wide experience library after T broadcast rounds satisfies:*

$$\mathbb{E}\left[\bar{q}^{(T)}\right] \geq q^* - \varepsilon_{approx} - O\left(\sqrt{\frac{\log N}{T}}\right), \tag{21}$$

where q^* is the quality achievable with centralized aggregation of all honest experiences and $\varepsilon_{approx} = O(f/N)$ is the irreducible Byzantine contamination term.

Proof. The proof proceeds in two steps.

Step 1 (Gossip Convergence). By the epidemic gossip analysis [Demers et al., 1987], after T rounds with fanout k_{peers}, the fraction of nodes that have received any given honest experience is $1 - O(e^{-k_{\text{peers}}T/N})$. For $T = O(\log N/k_{\text{peers}})$, this is $1 - O(1/N)$.

Step 2 (Quality Improvement). At each node, DeltaSoup retains all honest experiences (Part 2 of Theorem 4) and excludes most Byzantine contributions (Part 1). Each new honest experience from a remote node increases the local library quality by at least its marginal contribution. The marginal contribution decays as $O(1/M)$ as the library grows (diminishing returns). Summing over T rounds and $h = N - f$ honest nodes, the cumulative improvement is $\Omega(\sqrt{hT})$, giving the $O(\sqrt{\log N/T})$ convergence term. □

6 Fleet Learning Dynamics

6.1 Experience Quality Metric

Definition 8 (Fleet Experience Quality)**.** *The fleet-wide experience quality at round T is:*

$$Q^{(T)} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{|\mathcal{L}_i^{(T)}|} \sum_{E_m \in \mathcal{L}_i^{(T)}} q_m \cdot \text{Sim}(E_m, \mathcal{T}_i), \tag{22}$$

where $\text{Sim}(E_m, \mathcal{T}_i)$ is the cosine similarity between the experience embedding $\boldsymbol{e}_m$ and the mean embedding of node i's recent task distribution $\mathcal{T}_i$.

$Q^{(T)}$ captures both the intrinsic quality of experiences (via q_m) and their relevance to each node's task distribution (via Sim). A fleet where nodes only receive experiences from irrelevant domains would have high q_m but low Sim, correctly receiving a low Q score.

Theorem 6 (Monotonic Fleet Improvement)**.** *Under Assumptions 1–2, if DeltaSoup's filter threshold τ_{filter} is set such that Byzantine experiences are excluded with probability $\geq 1 - \delta$ (as in Theorem 4), then the expected fleet quality is non-decreasing:*

$$\mathbb{E}\left[Q^{(T+1)}\right] \geq \mathbb{E}\left[Q^{(T)}\right] - O(\delta). \tag{23}$$

Proof. At round $T + 1$, each node receives honest experiences from its peers and merges them via DeltaSoup. Any honest experience E_m with $\text{Sim}(E_m, \mathcal{T}_i) > 0$ and $q_m > 0$ increases the quality of node i's library, contributing positively to $Q^{(T+1)}$. Byzantine experiences that pass the filter (probability $\leq \delta$ per experience) may decrease quality. Taking expectations and applying linearity, the expected quality change is $\geq -O(\delta) \cdot |\mathcal{E}_{\text{Byzantine}}|$. Since $|\mathcal{E}_{\text{Byzantine}}| = O(f)$ and $f < N/3$, the net expected change is non-negative for sufficiently small δ. □

6.2 Information Flow Model

The fleet learning cycle for a single round is:

$$\underbrace{\text{Agent Session}}_{\text{tool calls, outcomes}} \xrightarrow{\text{GRPO}} \underbrace{\text{Local Experiences}}_{\mathcal{L}_i^{(T)}} \xrightarrow{\text{BitDelta}} \underbrace{\text{Compressed Batch}}_{\hat{B}_i} \xrightarrow{\text{Gossip}} \underbrace{\text{Peer Receives}}_{\hat{B}_j, j \neq i} \xrightarrow{\text{DeltaSoup}} \underbrace{\text{Updated Li}}_{\mathcal{L}_j^{(T+1)}} \tag{24}$$

Compression-Decompression Semantic Preservation. The information flow (24) introduces two lossy steps: (i) BitDelta compression (Theorem 2) and (ii) DeltaSoup filtering and merging. By Corollary 3, BitDelta preserves cosine similarity ≥ 0.95. The DeltaSoup merge (Eq. 19) is a convex combination within the cluster, so the merged embedding remains close to all cluster members. Combined, the end-to-end semantic preservation satisfies:

$$\cos(\boldsymbol{e}_m, \boldsymbol{e}_m^{\text{final}}) \geq 0.95 \cdot 0.95 = 0.9025, \tag{25}$$

where $\boldsymbol{e}_m^{\text{final}}$ is the embedding after compression, broadcast, decompression, and DeltaSoup merge.

6.3 Specialization vs. Generalization

Domain-Aware Routing. Each experience carries a domain tag (e.g., `coding`, `writing`, `data-analysis`). During broadcast, nodes may optionally subscribe to specific domain tags, filtering incoming batches to relevant domains only. This provides two benefits:

1. **Bandwidth reduction:** A coding agent need not receive marketing-domain experiences.
2. **Quality improvement:** Domain-filtered libraries have higher mean $\text{Sim}(E_m, \mathcal{T}_i)$, directly improving $Q^{(T)}$.

Fleet Diversity. Despite domain specialization, the fleet benefits from cross-domain transfer when tasks span multiple domains. DeltaSoup's confidence-weighted merge naturally weights domain-relevant experiences more heavily (as relevant experiences tend to have higher q_m from the receiving node's evaluations).

Proposition 7 (Specialization-Generalization Tradeoff). *Let $\rho \in [0, 1]$ be the fraction of received experiences that are in the node's primary domain. The fleet quality satisfies:*

$$Q^{(T)}(\rho) = \rho \cdot Q^{(T)}_{specialized} + (1 - \rho) \cdot Q^{(T)}_{general}, \tag{26}$$

where $Q^{(T)}_{specialized} \geq Q^{(T)}_{general}$ for single-domain tasks and $Q^{(T)}_{specialized} \leq Q^{(T)}_{general}$ for multi-domain tasks. The optimal ρ^* depends on task distribution breadth.

7 Integration with the Hanzo Agent Harness

7.1 The Continuous Learning Extension

FAI is implemented as a *plugin* for the Hanzo agent harness, augmenting the base agent with four capabilities:

1. **Session telemetry collection**: Intercept and log all tool calls and outcomes without modifying agent behavior.
2. **Background GRPO**: Run experience extraction asynchronously between sessions, not blocking agent execution.
3. **DSO network tools**: Expose DSO operations as callable tools so the agent can query its own fleet status.
4. **Periodic synchronization**: Background service that runs library consolidation and DSO broadcast on a configurable schedule.

7.2 DSO Network Tools

The plugin exposes four tools to the agent:

```
dso_network(action="status")
  -> { peers: N, library_size: M, last_sync: timestamp,
       bytes_sent: X, bytes_received: Y }

dso_network(action="broadcast", domain="coding")
  -> { experiences_sent: K, peers_reached: P, batch_id: str }

dso_network(action="receive", domain="coding", limit=50)
  -> { experiences_received: K, library_delta: D }

dso_network(action="peers")
  -> { peers: [{ node_id, latency_ms, reputation, last_seen }] }
```

These tools allow the agent to actively manage its DSO participation—for example, by broadcasting immediately after completing a difficult task, or by requesting experiences from a specific domain before starting a new project.

7.3 Experience Lifecycle

The complete lifecycle of an experience from agent session to fleet benefit is illustrated in Figure 1 and described below:

1. **Session**: Agent executes tools, producing telemetry $\{(\text{tool}_t, \text{input}_t, \text{output}_t, r_t)\}$.
2. **Trajectory construction**: Telemetry is grouped into trajectories $\tau = (\tau_1, \ldots, \tau_T)$ by task.
3. **GRPO extraction**: Group relative advantages are computed and semantic patterns are extracted via LLM introspection.
4. **Local library update**: Patterns are embedded and stored in $\mathcal{L}_i$, indexed by task embedding.
5. **BitDelta compression**: Library is compressed using Algorithm 1 with the current baseline.
6. **DSO broadcast**: Compressed batch is broadcast to k peers via the gossip protocol.
7. **Peer receive**: Peers receive the batch, verify integrity, and add to their incoming queue.
8. **DeltaSoup merge**: Algorithm 3 merges all received batches into the peer's local library.
9. **Better next session**: The updated library is used for PoE decoding (Eq. 2) in the peer's next session.

$$\begin{tikzcd}
\textbf{Session} \arrow[r] & \textbf{Telemetry} \arrow[r] & \textbf{GRPO} \arrow[r] & \begin{array}{c}\textbf{Local}\\ \mathcal{L}_i\end{array} \arrow[r] & \begin{array}{c}\textbf{Compress}\\ \text{BitDelta}\end{array} \\
 & \begin{array}{c}\textbf{Better Session}\\ \text{(PoE decoding)}\end{array} & \begin{array}{c}\textbf{DeltaSoup}\\ \text{merge}\end{array} \arrow[l] & \begin{array}{c}\textbf{Receive}\\ \text{peers}\end{array} \arrow[l] & \begin{array}{c}\textbf{Broadcast}\\ \text{DSO gossip}\end{array} \arrow[l]
\end{tikzcd}$$

Figure 1: Experience lifecycle: from agent session to fleet benefit and back.

7.4 Interaction with Self-Improvement

The Hanzo agent harness includes a self-improvement loop that periodically audits agent behavior and proposes policy changes. FAI integrates with this loop at two points:

1. **Library statistics inform sharing policy**: The self-improvement loop's maintenance phase (Loop 4) examines experience library statistics (domain coverage, quality distribution, staleness) and can increase broadcast frequency for under-represented domains or reduce it for saturated ones.
2. **Fleet quality informs individual improvement**: Low fleet-wide quality in a domain signals that new local GRPO runs may be productive. The self-improvement loop uses $Q^{(T)}$ as a signal for when to invest in fresh rollouts.

8 Privacy and Security

8.1 What Is and Is Not Shared

Shared (non-sensitive):

- Natural language pattern descriptions (≤ 32 words each), representing general strategies (e.g., "validate inputs before calling external APIs").
- 1-bit quantized embeddings of these patterns.
- Quality scores and domain tags.

Never shared:

- Raw tool call inputs or outputs (may contain user data).
- Agent trajectories or session logs.
- Model weights or internal representations.
- User identifiers, API keys, or credentials.
- Task-specific content beyond the extracted pattern.

The pattern extraction step (GRPO introspection) is designed to produce generalizable principles, not task-specific solutions. A pattern like "always handle `None` inputs before processing" does not reveal what specific `None` value was encountered.

8.2 Differential Privacy

For deployments requiring formal privacy guarantees, FAI supports optional Gaussian noise injection before broadcast:

Definition 9 (DP-Noisy Compression)**.** *Given privacy budget $\varepsilon_{DP} > 0$ and sensitivity Δ_s, add Gaussian noise to each scale factor before broadcast:*

$$\tilde{\alpha}_m^{(j)} = \alpha_m^{(j)} + \mathcal{N}\big(0,\, \sigma_{DP}^2\big)\,, \quad \sigma_{DP} = \frac{\Delta_s\sqrt{2\ln(1.25/\delta)}}{\varepsilon_{DP}}. \tag{27}$$

Sign vectors are not perturbed (binary data has a different privacy calculus). Each broadcast round consumes privacy budget ε_{DP}; over R rounds, the total budget is $R \cdot \varepsilon_{DP}$ (basic composition) or $O(\varepsilon_{DP}\sqrt{R\ln(1/\delta)})$ (advanced composition [Dwork & Roth, 2014]).

8.3 Integrity and Reputation

Batch Integrity. Each broadcast batch is signed by the sender's private key (using Ed25519) and includes a SHA-256 hash. Receiving nodes verify both before processing.

Reputation Tracking. Each node maintains a reputation score for each peer, initialized to 1.0 and updated as:

$$\mathrm{rep}_i^{(T+1)} = \gamma \cdot \mathrm{rep}_i^{(T)} + (1-\gamma) \cdot r_i^{(T)}, \tag{28}$$

where $\gamma = 0.9$ is the decay factor and $r_i^{(T)} \in \{0, 1\}$ indicates whether peer i's batch passed validation (hash, freshness, and filter). Peers with $\mathrm{rep}_i < \rho_{\min}$ (default $\rho_{\min} = 0.3$) are deprioritized for future gossip connections.

DSO Stake Requirement. Integration with the DSO network requires staking a minimum bond $D_{\min}$ (default: 10 $AI) per node registration. Byzantine behavior that is provably detected triggers bond slashing, creating an economic deterrent against adversarial participation [Hanzo AI Research, 2026b].

9 Experimental Evaluation

9.1 Simulation Setup

We evaluate FAI via a discrete-event simulator implementing the full protocol stack. Each simulated agent node runs local GRPO and DSO on a fixed task distribution drawn from SWE-bench Verified [Jimenez et al., 2024] (software engineering tasks).

Network Configurations. We evaluate three fleet sizes: $N \in \{10, 50, 100\}$ nodes. For each, we fix gossip fanout $k_{\text{peers}} = 8$, broadcast interval 60 s, group size $g = 128$, and filter threshold $\tau_{\text{filter}} = 2.0$.

Baselines.

- **Isolated ASO**: Each node runs ASO independently with no sharing. Represents the single-agent baseline.
- **Centralized DSO**: All experiences are aggregated at a central server (no Byzantine robustness, no compression). Represents the oracle upper bound.
- **FAI (ours)**: Full federated protocol with BitDelta and DeltaSoup.
- **FAI (no DeltaSoup)**: FAI without Byzantine-robust filtering, using naive averaging instead.

9.2 Task Success Rate Results

Method	Fleet Size (N)			Improvement over Isolated
	$N = 10$	$N = 50$	$N = 100$	
Isolated ASO	18.2%	18.2%	18.2%	–
Centralized DSO	22.1%	23.4%	24.0%	up to +5.8%
FAI, no DeltaSoup	20.1%	21.3%	21.9%	up to +3.7%
FAI (ours)	**21.8%**	**22.9%**	**23.7%**	up to +**5.5**%

Table 1: Task success rate (SWE-bench resolved rate) by method and fleet size, with 0% Byzantine nodes. FAI approaches the centralized oracle while maintaining full decentralization. DeltaSoup provides +1.7% improvement over naive averaging.

9.3 Byzantine Robustness

Method	$f = 0$	$f = N/10$	$f = N/4$	$f = N/3$
FAI, no DeltaSoup	23.7%	21.8%	19.4%	18.1%
FAI (ours)	**23.7%**	**23.1%**	**22.8%**	**22.0%**

Table 2: Task success rate at $N = 100$ with varying Byzantine fraction. DeltaSoup maintains 92.8% of clean performance at $f = N/3$, while naive averaging degrades to isolated-agent performance at $f = N/3$.

9.4 Compression Efficiency

Representation	Bytes/experience	Total ($M = 100$)	Compression
Float32 embedding ($d = 768$)	3,072	307,200	1×
BitDelta ($g = 128$)	288	28,800	10.7×
BitDelta + metadata	352	35,200	8.7×

Table 3: Storage and bandwidth costs per experience and per batch. The "+ metadata" row includes text pattern (avg. 128 bytes), quality score (4 bytes), domain tag (8 bytes), and node ID (32 bytes).

9.5 Cosine Similarity Preservation

Group size g	Mean cosine sim.	Min cosine sim.	Compression ratio
1	0.704	0.582	32×
16	0.861	0.793	16×
64	0.934	0.901	12.1×
128	**0.957**	**0.931**	10.7×
256	0.971	0.952	9.1×
Uncompressed	1.000	1.000	1×

Table 4: Cosine similarity between original and reconstructed embeddings by group size, averaged over 1,000 experiences from a sentence-transformer encoder. Group size $g = 128$ provides the best compression-fidelity tradeoff.

9.6 Convergence Speed

Fleet Size	Rounds to 90% of Centralized	Final Quality ($T = 100$)	Bandwidth/Round
$N = 10$	12	98.6% of centralized	29 KB/node
$N = 50$	18	97.9% of centralized	29 KB/node
$N = 100$	23	98.8% of centralized	29 KB/node

Table 5: Convergence speed and final quality relative to the centralized oracle. Bandwidth per node per round is constant regardless of fleet size (gossip fanout $k = 8$, 100 experiences, BitDelta compressed).

10 Discussion and Future Work

PoAI Integration. Proof-of-AI (ZIP-002) [Hanzo AI Research, 2026d] provides verifiable attestation that experiences were genuinely extracted from real agent trajectories, not fabricated. Integrating PoAI with FAI would provide a cryptographic certification layer, allowing nodes to assign higher quality scores to PoAI-attested experiences. This addresses the scenario where Byzantine nodes submit syntactically valid but semantically fabricated patterns that pass the DeltaSoup filter.

Hierarchical DSO for Large Fleets. For very large fleets ($N > 1000$), flat gossip broadcast becomes bandwidth- intensive. A hierarchical protocol organizes nodes into clusters (cluster $\rightarrow$ region $\rightarrow$ global), with DeltaSoup applied at each level:

$$\mathcal{L}_{\text{global}} = \textsc{DeltaSoup}(\{\textsc{DeltaSoup}(\{\mathcal{L}_{i,r}\}_{i \in R}) : r \in \mathcal{R}\}) , \tag{29}$$

where $\mathcal{R}$ is the set of regions and $\{i \in R\}$ denotes nodes in region R. The two-level Byzantine robustness guarantees are preserved as long as each region has $f < N_r/3$ Byzantine nodes.

Economic Incentives for Experience Sharing. High-quality experience contributions should be rewarded to incentivize participation. An integration with the DSO token economy would issue \$AI tokens to nodes whose contributed experiences (as measured by the improvement they produce in receiving nodes' task success rates) exceed a quality threshold. This requires a mechanism for attributing outcome improvement to specific received experiences—a causal inference problem that remains open.

Cross-Domain Transfer via Embedding Alignment. Currently, domain filtering restricts cross-domain sharing. For tasks that span multiple domains (e.g., a coding task that also requires documentation), embedding alignment techniques [Conneau et al., 2020] could map experiences from different domains into a shared semantic space, enabling more flexible knowledge transfer.

Online GRPO. The current system runs GRPO in batch mode after sessions conclude. An online variant would update the experience library in real time during agent execution, allowing rapid adaptation within a single long session. This requires careful management of the GRPO group buffer and may interact with active DeltaSoup merges.

Adversarial Pattern Detection. DeltaSoup's geometric median filter catches embedding-space adversaries but may not detect experiences with valid embeddings that carry subtle misinformation in their natural language text (e.g., patterns that appear useful but subtly suggest insecure practices). Future work should incorporate semantic consistency checks: verifying that the pattern text is consistent with the embedding, and that applying the pattern on held-out tasks does not degrade performance.

11 Related Work

Federated Learning. FedAvg [McMahan et al., 2017] and FedProx [Li et al., 2020] aggregate model parameters; FedLoRA [Zhang et al., 2024] federates LoRA adapters. All require gradient computation and access to model weights. FAI requires neither, sharing only semantic experiences extracted from black-box model outputs.

Byzantine-Robust Distributed Learning. Krum [Blanchard et al., 2017] selects the most consistent gradient. Yin et al. [2018] analyze coordinate-wise median and trimmed mean. Bulyan [El Mhamdi et al., 2018] and DRACO [Chen et al., 2018] provide stronger guarantees. DeltaSoup adapts coordinate-wise median to the semantic embedding setting, where dimensions have semantic meaning and group-wise MAD provides a more principled filter than a fixed threshold.

Multi-Agent Experience Sharing. Baker et al. [2020] study emergent tool use through shared replay buffers in model-based RL. Christianos et al. [2021] analyze selective parameter sharing for multi-agent RL. FAI differs by sharing compressed semantic priors rather than raw experience tuples or model parameters, and by operating on frozen LLMs rather than trainable policies.

Communication-Efficient Distributed Learning. 1-bit SGD [Seide et al., 2014] and SignSGD [Bernstein et al., 2018] quantize gradients. Top-k sparsification [Stich et al., 2018] compresses by transmitting only the largest-magnitude gradient elements. BitDelta [Hanzo AI Research, 2026c] adapts 1-bit quantization to the model weight delta setting; we further adapt it to the experience embedding setting where the delta is from the library baseline rather than a previous model checkpoint.

Decentralized AI Networks. Bittensor [Bittensor, 2023] creates a marketplace for ML model outputs. Gensyn [Gensyn, 2023] provides verifiable distributed training. DSO and FAI differ by focusing on post-training adaptation (no gradient computation) and targeting the experience—not the model—as the shared artifact.

Knowledge Distillation. Born-again networks [Furlanello et al., 2018] and knowledge distillation [Hinton et al., 2015] transfer knowledge between models via soft labels. FAI transfers knowledge via natural language patterns and compressed embeddings, which are interpretable, shareable, and storable without requiring teacher-student model pairs.

Privacy-Preserving Federated Learning. Differential privacy for federated learning [McMahan et al., 2018] adds calibrated noise to model updates. Secure aggregation [Bonawitz et al., 2017] protects individual updates via cryptographic protocols. FAI supports optional DP noise injection (Definition 9) and inherits the DSO network's hash-based integrity.

12 Conclusion

We presented **Federated Agent Intelligence (FAI)**, a system enabling collective learning across autonomous AI agent fleets without sharing model weights, raw data, or training compute. The core insight is that frozen language model agents learn implicitly through experience, and that this knowledge can be extracted as compressed semantic priors, shared via decentralized gossip, and aggregated robustly against Byzantine adversaries.

FAI composes three mechanisms: GRPO-based local experience extraction, BitDelta compression achieving $\approx$10$\times$ bandwidth reduction with $< 5\%$ semantic loss, and DeltaSoup Byzantine-robust aggregation tolerating $f < N/3$ adversarial contributors. We proved formal guarantees including convergence rate $O(\sqrt{\log N/T})$, (f, ε)-Byzantine robustness, and monotonic fleet quality improvement under honest majority.

Key Results.

- **23.7% task success rate** at $N = 100$ nodes, versus 18.2% isolated (a +5.5% absolute improvement approaching the centralized oracle's +5.8%).
- **Byzantine resilience**: 92.8% of clean performance retained at $f = N/3$ with DeltaSoup; naive averaging collapses to baseline.
- 10.7$\times$ **compression**: BitDelta reduces per-experience bandwidth from 3,072 bytes to 288 bytes, enabling practical P2P operation.
- **0.957 mean cosine similarity** after compression/decompression cycle at group size $g = 128$.
- **Constant per-node bandwidth**: $\approx$29 KB/round regardless of fleet size (gossip fanout $k = 8$).

Limitations. DeltaSoup requires $f < N/3$; larger Byzantine fractions degrade performance. The experience quality metric (Definition 8) is a proxy for downstream task performance and may not perfectly correlate in all domains. Cross-domain transfer is limited by domain tag filtering.

Broader Impact. FAI enables organizations to share the benefits of agent experience without sharing sensitive data. Any agent operating in a domain benefits from the collective experience of all agents in that domain, accelerating the time-to-competence for new deployments and reducing redundant exploration across the fleet. This democratizes access to high-quality AI agent performance for smaller organizations that cannot afford to run large independent fleets.

A Proof of Coordinate-Wise Median Robustness

Theorem 8 (Robustness of Coordinate-Wise Median). *Let $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ with $f < N/2$ adversarial values and $h = N - f$ honest values drawn from distribution $\mathcal{D}$ with mean μ and coordinate-wise standard deviation σ_j. The coordinate-wise median $\hat{\mu}_j = \mathrm{med}(\{x_{i,j}\}_{i=1}^N)$ satisfies:*

$$|\hat{\mu}_j - \mu_j| \leq O\left(\frac{f}{h} \cdot \sigma_j\right) + O\left(\frac{\sigma_j}{\sqrt{h}}\right) \qquad \forall j \in [d], \tag{30}$$

with probability $\geq 1 - 2\exp(-h/8)$.

Proof. Fix coordinate j. Let F_h be the empirical CDF of the h honest values $\{x_{i,j}\}_{i \in H}$. The true median $m_j = F_h^{-1}(1/2)$ satisfies $|m_j - \mu_j| = O(\sigma_j/\sqrt{h})$ by the CLT for order statistics.

The adversary can shift the sample median by at most f positions in the sorted order. The median of $N = h + f$ values corresponds to the $(N/2)$-th order statistic. With f adversarial values placed optimally, the sample median corresponds to at most the $(h/2 + f)$-th honest order statistic, i.e., the $((h/2 + f)/h)$-th quantile of the honest distribution. Since $f < h/2$ (implied by $f < N/3$), this is the $(1/2 + f/h)$-th quantile. By the quantile deviation bound: $|F_h^{-1}(1/2 + f/h) - F_h^{-1}(1/2)| \le (f/h) \cdot \sigma_j/p_j$, where p_j is the density of the honest distribution at m_j. Under sub-Gaussianity, $p_j = O(1/\sigma_j)$, giving $O(f\sigma_j/h)$. Combining with the honest-median deviation completes the proof. $\square$

B DeltaSoup: Detailed Filter Analysis

Proposition 9 (Filter False Positive Rate)**.** *Under Assumption 2, the fraction of honest experiences filtered by the MAD filter (Eq. 18) with threshold* $\tau = 2$ *is bounded by:*

$$P(\textit{honest filtered}) \le \frac{d \cdot \mathrm{Var}[|\Delta|]}{4\sigma_h^2}. \tag{31}$$

For well-clustered libraries with $\mathrm{Var}[|\Delta|] \ll \sigma_h^2$*, this is negligible.*

Proof. An honest experience $\hat{\boldsymbol{e}}_m$ is filtered when its mean normalized deviation exceeds $\tau = 2$. By Markov's inequality: $P(\bar{D}_m > 2) \le \bar{D}_m/2 \le d \cdot \mathrm{Var}[|\Delta|]/(4\sigma_h^2)$, where we used $\mathbb{E}[\bar{D}_m] = d \cdot \mathrm{Var}[|\Delta|]/\sigma_h^2$ (the mean deviation of honest values from the coordinate-wise median is proportional to the variance of $|\Delta|$ relative to σ_h). $\square$

C Bandwidth Analysis

The per-node bandwidth consumption per round is:

Outgoing bandwidth:

$$B_{\text{out}} = k_{\text{peers}} \cdot |B_{\text{compressed}}| = k_{\text{peers}} \cdot M \cdot \left(\frac{d}{8} + \frac{d}{g} \cdot 4 + 64\right) \text{ bytes}. \tag{32}$$

For $k_{\text{peers}} = 8$, $M = 100$, $d = 768$, $g = 128$: $B_{\text{out}} = 8 \times 29{,}472 \approx 236\,\text{KB/round}$.

Incoming bandwidth: Each node receives batches from k_{peers} peers, so $B_{\text{in}} \approx B_{\text{out}}$.

Gossip propagation: A single experience reaches all N nodes within $T_{\text{prop}} = O(\log N/\log k_{\text{peers}})$ rounds [Demers et al., 1987]. For $N = 100$, $k = 8$: $T_{\text{prop}} \approx 2.2$ rounds.

D Reproducibility Details

- **Environment:** Python 3.11, LanceDB 0.3.0, sentence-transformers 1.2.0 (all-MiniLM-L6-v2, $d = 384$; we use $d = 768$ in the paper via all-mpnet-base-v2).
- **Network simulation:** Discrete-event simulator with configurable latency (uniform 10–100 ms), packet loss ($p = 0.01$), and node failures.

- **Byzantine behavior:** Random noise ($\boldsymbol{s}_{i,m}^{(j)} \sim \text{Unif}(\{-1, +1\}^g)$, $\alpha_{i,m}^{(j)} \sim \text{Unif}(0, 1)$).
- **GRPO setup:** Group size $G = 4$, max iterations $K = 3$, reward weights $\alpha = 0.7$, $\beta = 0.3$.
- **Seeds:** Fixed seeds 42, 123, 456 for three independent runs; all tables report mean ± std ($\leq 0.4\%$ in all cases).
- **Hardware:** Simulated on a single 16-core x86 machine; no GPU required for simulation.

References

Baker, B., Kanitscheider, I., Marber, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I. Emergent tool use from multi-agent autocurricula. *ICLR*, 2020.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signSGD: Compressed optimisation for non-convex problems. *ICML*, 2018.

Bittensor. Bittensor: A peer-to-peer intelligence market. Whitepaper, 2023.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. *NeurIPS*, 2017.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. *ACM CCS*, 2017.

Chen, L., Wang, H., Charles, Z., and Papailiopoulos, D. DRACO: Byzantine-resilient distributed training via redundant gradients. *ICML*, 2018.

Christianos, F., Papoudakis, G., Rahman, M. A., and Albrecht, S. V. Scaling multi-agent reinforcement learning with selective parameter sharing. *ICML*, 2021.

Conneau, A., Rinott, R., Lample, G., Williams, A., Bowman, S., Schwenk, H., and Stoyanov, V. XNLI: Evaluating cross-lingual sentence representations. *EMNLP*, 2020.

Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., and Terry, D. Epidemic algorithms for replicated database maintenance. *PODC*, 1987.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Furlanello, T., Lipton, Z. C., Tschannen, M., Itti, L., and Anandkumar, A. Born again neural networks. *ICML*, 2018.

Gensyn. Gensyn: Protocol for decentralized ML computation. Whitepaper, 2023.

Hanzo AI Research. Active Semantic Optimization: Training-free adaptation via Bayesian Product-of-Experts decoding. Technical report, Hanzo AI, 2026.

Hanzo AI Research. Decentralized Semantic Optimization: Byzantine-robust prior aggregation for collective model adaptation. Technical report, Hanzo AI (ZIP-001/400), 2026.

Hanzo AI Research. BitDelta: 1-bit quantization of experience embedding deltas for efficient fleet communication. Technical report, Hanzo AI (ZIP-007), 2026.

Hanzo AI Research. Proof of AI: Verifiable attestation for agent trajectory extraction. Technical report, Hanzo AI (ZIP-002), 2026.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *NeurIPS Workshop*, 2015.

Jiménez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. SWE-bench: Can language models resolve real-world GitHub issues? *ICLR*, 2024.

Koloskova, A., Stich, S. U., and Jaggi, M. Decentralized stochastic optimization and gossip algorithms with compressed communication. *ICML*, 2020.

Lamport, L., Shostak, R., and Pease, M. The Byzantine generals problem. *ACM TOPLAS*, 4(3):382–401, 1982.

LanceDB. LanceDB: Serverless vector database for AI applications. https://lancedb.com, 2024.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *MLSys*, 2020.

Liu, J., Kulikov, R., Fox, Z., Nikdan, A., Kim, Y., Papailiopoulos, D., and De Sa, C. BitDelta: Your fine-tune may only be worth one bit. *arXiv preprint arXiv:2402.10193*, 2024.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. *AISTATS*, 2017.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. *ICLR*, 2018.

El Mhamdi, E. M., Guerraoui, R., and Rouault, S. The hidden vulnerability of distributed learning in Byzantium. *ICML*, 2018.

Minsker, S. Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335, 2015.

Pease, M., Shostak, R., and Lamport, L. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

Seide, F., Fu, H., Droppo, J., Li, G., and Yu, D. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs. *Interspeech*, 2014.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Stich, S. U., Cordonnier, J.-B., and Jaggi, M. Sparsified SGD with memory. *NeurIPS*, 2018.

Warnat-Herresthal, S., Schultze, H., Shastry, K. L., et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature*, 594:265–270, 2021.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. *ICML*, 2018.

Zhang, J., Hua, Y., Wang, H., Song, T., Xue, Z., Ma, R., and Guan, H. FedLoRA: When personalized federated learning meets low-rank adaptation. *arXiv preprint arXiv:2307.06768*, 2024.