

# Zoo DSO: Decentralized Semantic Optimization with Byzantine-Robust Prior Aggregation

Zach Kelling\*

*Hanzo Industries   Lux Industries   Zoo Labs Foundation*  
research@lux.network

October 2025

## Abstract

We present **Zoo DSO** (Decentralized Semantic Optimization), a protocol developed by Zoo Labs Foundation that extends Hanzo’s ASO (HIP-002) with decentralized prior aggregation. DSO enables zero-training model adaptation by (i) extracting semantic advantages via Hanzo’s TF-GRPO, (ii) compressing priors with 1-bit quantization ( $29.5 \times$  savings), (iii) aggregating contributions via byzantine-robust median voting, and (iv) synchronizing via IPFS/Arweave with on-chain registry verification. Key contributions: (a) Byzantine-robust aggregation scheme with stake-weighted quality scores, (b) Content-addressed experience registry with Merkle proofs, (c) Gossip-based P2P sync optimized for high-quality shard propagation, and (d) Integration with Hanzo Network’s PoAI consensus (HIP-004) for attestation-based rewards. We demonstrate **15.2% improvement** in multi-agent code generation tasks when nodes share priors vs. isolated operation. This work builds on years of collaboration between Zoo Labs Foundation and Hanzo AI Inc.

## 1 Introduction

Traditional federated learning aggregates gradients or parameters, requiring compute-intensive operations and risking catastrophic interference. Recent work on parameter-efficient adaptation (LoRA, adapters) reduces overhead but still requires training. In-context learning enables zero-shot adaptation but lacks systematic mechanisms for sharing knowledge across agents.

**Our Contribution.** Zoo DSO extends Hanzo’s Active Semantic Optimization (ASO, HIP-002) with a decentralized protocol for sharing *experiential priors*—compressed semantic advantages extracted from agent rollouts—enabling collective intelligence without parameter updates. While Hanzo provides the base infrastructure (TF-GRPO, PoE, HMM, PoAI), Zoo adds Byzantine-robust consensus and decentralized storage for multi-agent scenarios.

## 2 System Architecture

### 2.1 Roles and Components

- **Agents:** Run local ASO (TF-GRPO), extract and compress priors

---

\*zach@lux.network

- **Registry:** On-chain smart contract storing CIDs, Merkle roots, quality scores
- **Storage:** Off-chain IPFS/Arweave for prior data
- **Aggregators:** Compute byzantine-robust median under fixed schema
- **Validators:** Verify attestations via PoAI, slash malicious submissions

### 3 Byzantine-Robust Aggregation

#### 3.1 Threat Model

Adversaries may submit:

- **Random noise:** Low-quality priors to pollute aggregate
- **Targeted attacks:** Priors designed to degrade specific tasks
- **Sybil attacks:** Multiple identities voting for malicious priors

#### 3.2 Median Voting Under Schema

Fix a schema  $\mathcal{S}$ : token bins or embedding centroids. Each submission  $E_i$  is decomposed:

$$E_i = \{\Delta_i^{(s)}\}_{s \in \mathcal{S}}, \quad (1)$$

where  $\Delta_i^{(s)}$  is the advantage for schema element  $s$ . The aggregate computes:

$$\tilde{E}^{(s)} = \text{weighted-median}_i(\Delta_i^{(s)}; w_i), \quad (2)$$

with weights  $w_i$  proportional to stake times quality score  $q_i$ .

#### 3.3 Quality Scoring

Quality scores  $q_i \in [0, 1]$  are estimated via:

1. **Holdout validation:** Test priors on reserved benchmarks
2. **Cross-validation:** Peer nodes sample and verify
3. **Reputation:** Historical performance on-chain

Quality below threshold  $q_{\min}$  (default 0.3) triggers bond slashing.

### 4 ExperienceRegistry Contract

#### 4.1 Interface

```
interface IExperienceRegistry {
    struct Entry {
        bytes32 merkleRoot;
        string cid;           // IPFS/Arweave
        uint64 schema;
```

```

    uint64 quality;      // quantized [0, 1000]
    address submitter;
    uint256 bond;
}
function submit(Entry calldata e) external payable
    returns (uint256 id);
function voteQuality(uint256 id, uint64 score) external;
function slash(uint256 id, address challenger,
    bytes calldata proof) external;
function aggregate(uint256[] calldata ids) external
    returns (bytes32 aggregateMerkleRoot);
}

```

## 4.2 Bond and Slashing

Submission requires bond  $D$  (default 25 \$AI). If  $q_i < q_{\min}$ , challenger submits proof (counter-examples); successful challenge burns  $\sigma D$  (default  $\sigma = 0.5$ ), refunds  $(1 - \sigma)D$  to challenger.

# 5 P2P Synchronization

## 5.1 Gossip Protocol

Nodes maintain local replica of high-quality priors:

1. Subscribe to registry events (new submissions, quality updates)
2. Fetch CID from IPFS/Arweave if  $q_i \geq q_{\text{fetch}}$  (default 0.5)
3. Verify Merkle proof against on-chain root
4. Merge into local LanceDB with CRDT semantics

## 5.2 Incentive Alignment

Nodes that propagate high-quality priors earn fee rebates:

$$\text{rebate}_i \propto \sum_{j:\text{fetched from } i} q_j \cdot \text{size}_j. \quad (3)$$

# 6 Integration with PoAI

## 6.1 Attestation-Based Rewards

Each prior submission includes PoAI attestation (TEE report + task metrics):

- $\Delta I$ : Information gain from experience
- $\Delta U$ : Utility improvement on held-out tasks
- Cost metrics: compute, bandwidth, energy

Emissions formula (see HMM paper):

$$R_i = \gamma\Delta I + \beta\Delta U - \lambda_c \cdot \text{cost}_i, \quad (4)$$

where  $\gamma = 1.0$ ,  $\beta = 0.5$ ,  $\lambda_c = 0.1$  (defaults).

## 7 Experimental Evaluation

### 7.1 Multi-Agent Code Generation

Configuration	Resolved Rate	Avg. Prior Reuse
Isolated agents (no DSO)	16.3%	0.0
DSO (Byzantine honest)	18.8%	3.2 priors/task
DSO (20% Byzantine)	17.9%	2.8 priors/task
<b>DSO (median voting)</b>	<b>18.7%</b>	<b>3.1 priors/task</b>

Table 1: SWE-bench Verified (500 issues), 10 agents. Byzantine nodes submit random priors.

### 7.2 Storage Efficiency

- Full-precision priors: 2.4 GB/agent
- 1-bit compressed: 82 MB/agent ( $29.3\times$  savings)
- IPFS overhead: +12% (metadata, proofs)
- Effective:  $\approx 26\times$  savings

## 8 Related Work

**Federated learning:** FedAvg, FedProx, byzantine-robust aggregation. **Decentralized ML:** Swarm learning, peer-to-peer training. **Parameter-efficient adaptation:** LoRA, adapters, prompt tuning. **Blockchain + ML:** Federated learning on blockchain, decentralized model markets.

## 9 Conclusion

Zoo DSO enables decentralized, zero-training model adaptation by sharing compressed experiential priors across distributed agents. Built on Hanzo’s ASO foundation (HIP-002) and integrated with Hanzo Network’s economic layer (HIP-004), DSO adds Byzantine-robust aggregation to ensure resilience against adversarial nodes. This represents **years of co-development** between Zoo Labs Foundation and Hanzo AI Inc, with Zoo focusing on decentralized learning while Hanzo provides the base infrastructure. Future work includes cross-domain transfer (code  $\rightarrow$  data science) and hierarchical aggregation (specialized sub-groups).

## Acknowledgments

This work is the result of collaborative research between Zoo Labs Foundation (501c3) and Hanzo AI Inc (Techstars '17). Zoo extends Hanzo's Active Semantic Optimization (HIP-002) with Byzantine-robust consensus and integrates with Hanzo Network's Hamiltonian Market Maker (HIP-004) for economic incentives.

## A Security Analysis

### A.1 Sybil Resistance

Stake-weighting limits influence of Sybil identities. With  $N$  honest nodes and  $M < N/2$  Sybil identities (each with stake  $s$ ), median voting ensures honest aggregate if total honest stake  $\geq$  total malicious stake.

### A.2 Data Poisoning

1-bit quantization limits information content per prior, reducing attack surface. Median voting filters extreme values. Quality scoring enables post-hoc detection.

*Disclaimer.* This document describes a proposed protocol. Security properties require formal verification.