Hanzo Network: A Hamiltonian Market Maker Layer-1 for Decentralized AI Compute and Semantic Learning

Hanzo Industries Inc.
995 Market St, San Francisco, CA
contact@hanzo.ai

October 2025

Abstract

We present *Hanzo Network*, a specialized Layer-1 (L1) blockchain for AI compute exchange and decentralized semantic learning. Hanzo introduces a **Hamiltonian Market Maker (HMM)**—a provably-stable, oracle-minimal automated market maker that prices heterogeneous compute resources via a Hamiltonian invariant. Hanzo's semantic learning layer executes **Decentralized Semantic Optimization (DSO)** and **Active Semantic Optimization (ASO)**: training-free adaptation that shares token/embedding-level *experiential priors* across models and nodes. Contributions include: (i) an HMM with invariant $\mathcal{H}$ for multi-asset compute markets and continuous-time price dynamics, (ii) a *Proof of AI* (PoAI) consensus-extension for verifiable inference/training work, (iii) a Training-Free GRPO (TF-GRPO) scheme formalized as Bayesian product-of-experts (PoE) decoding, and (iv) a BitDelta-inspired 1-bit semantic compression enabling $29.5\times$ storage and multi-tenant serving efficiency. We detail protocols, security, and token economics for the $AI open protocol token used for staking, fees, rewards, and settlement.

1 Introduction

Modern AI systems are bottlenecked by (1) scarce, dynamic supply of compute (GPUs, memory, bandwidth), and (2) high-cost, siloed model adaptation. Hanzo addresses both by (a) making compute a first-class on-chain asset with a Hamiltonian AMM that clears resource markets without fragile oracles, and (b) providing decentralized, *zero-training* semantic learning (TF-GRPO) so all participants benefit from shared experiences without finetuning.

Vision. Hanzo Network integrates both the **compute L1** and **semantic learning layer** into a unified protocol. Nodes earn $AI by (i) providing compute and (ii) contributing high-quality experiential priors validated on-chain. The result is a transparent, efficient, and privacy-preserving substrate where any LLM can improve using cross-LLM experiences while jobs are priced and cleared in real time.

2 Design Goals

- **Oracle-minimal pricing:** resource prices arise endogenously from a conservative invariant; external price feeds are optional.
- **Verifiable work:** inference/training attestations via TEE-anchored receipts and/or succinct proofs.

- **Zero-training adaptation:** TF-GRPO & PoE decoding with compressed priors (1-bit deltas) for cheap personalization.
- **Byzantine robustness:** median-based aggregation for experiences; slashing for fraudulent attestations.
- **Composable L1:** EVM compatibility for DeFi/primitives; modules for markets, registry, staking.

3 System Overview

Roles. Validators; *Workers* (GPU, CPU, RAM, storage); *Routers* (batching/scheduling); *Curators* (experience quality signals). **Assets.** $C_{gpu}, C_{vram}, C_{ram}, C_{net}, C_{disk}$ (resource tokens); Q (demand credits); \$AI (settlement/staking). **Architecture.** Hanzo L1 hosts HMM, registry, DSO/ASO, and P2P sync for experiential priors.

4 Hamiltonian Market Maker (HMM)

4.1 Invariant and State

Let reserve vector $\boldsymbol{R} = (\Psi, \Theta)$ denote effective supply of compute capacity Ψ (e.g., GPU-seconds weighted by quality) and an aggregate demand credit pool Θ. A minimal HMM uses the **bilinear** Hamiltonian

$$\mathcal{H}(\Psi, \Theta) = \Psi\,\Theta = \kappa, \quad \kappa > 0, \tag{1}$$

which matches the constant-product AMM as a special case. For multi-asset resources $\boldsymbol{\Psi} = (\Psi_1, \ldots, \Psi_m)$ and credits $\boldsymbol{\Theta}$, we use

$$\mathcal{H}(\boldsymbol{\Psi}, \boldsymbol{\Theta}) = \sum_{i=1}^{m} w_i\,\Psi_i\,\Theta_i + \lambda \sum_{i=1}^{m} \tfrac{1}{2}(\Psi_i^2 + \Theta_i^2), \quad w_i, \lambda > 0. \tag{2}$$

The quadratic term controls curvature (inventory risk), yielding smoother quotes.

4.2 Prices, Flows, and Fees

Define the conjugate price for compute class i:

$$p_i \equiv \frac{\partial \mathcal{H}/\partial \Psi_i}{\partial \mathcal{H}/\partial \Theta_i} = \frac{w_i\,\Theta_i + \lambda\,\Psi_i}{w_i\,\Psi_i + \lambda\,\Theta_i}. \tag{3}$$

A swap $\Delta\boldsymbol{\Theta} < 0, \Delta\boldsymbol{\Psi} > 0$ (buy compute) preserves $\mathcal{H}$ up to fee f. We charge a split fee $f = f_m + f_r$: market fee f_m (LP/treasury) and *risk fee* $f_r \propto \|\Delta\boldsymbol{\Psi}\|$ to compensate inventory risk. In continuous time, inventory evolves via

$$\dot{\Psi}_i = s_i - u_i, \quad \dot{\Theta}_i = d_i - v_i, \quad \text{s.t. } \frac{d}{dt}\mathcal{H}(\boldsymbol{\Psi}, \boldsymbol{\Theta}) = 0 \,(\text{net of fees}) \tag{4}$$

with supply inflow s_i (workers) and demand d_i (jobs). Stability follows from convexity of $\mathcal{H}$ in each orthant and fee dissipation.

4.3 Composable Market Objects

Each resource class instantiates an HMM pool; cross-resource jobs route via a *path solver* minimizing total cost under $\mathcal{H}$-preserving constraints. Jobs specify an SLA vector (latency, jitter, region), encoded as Lagrange multipliers in the solver; quotes reflect SLA shadow prices.

5 Proof of AI (PoAI) and Job Settlement

5.1 Task Lifecycle

(1) Client escrows $AI and mints a credit $\Delta\Theta$. (2) Router clears against HMM to allocate $\Delta\Psi$. (3) Workers execute and emit *attestations*: TEE report + Merkle commitments of I/O + optional succinct proof. (4) Verifiers sample-check; (5) Settlement releases $AI to workers, rebates unused capacity to pool, distributes fees.

5.2 Attestation Primitives

TEE path: enclave measurements + signed runtime traces. *ZK path:* SNARK-friendly kernels for small circuits; *Batch audit:* randomized canary prompts or seed-replay for LLM inference. Misbehavior triggers slashing and denial windows.

6 Decentralized Semantic Optimization (DSO)

6.1 Experience Priors

Each agent/node maintains an *experience prior* E: token/embedding-level memory distilled from rollouts. Locally, nodes run **Active Semantic Optimization (ASO)** to extract *semantic advantages* from groups of rollouts (TF-GRPO). Priors are compressed (§7) and written to the on-chain *ExperienceRegistry* with Merkle proofs.

6.2 Training-Free GRPO as Bayesian PoE

For a base model with conditional $p_\theta(y \mid x)$ and a set of experiences $\{e_k\}$ mapping to token-level factors $\phi_k(y \mid x)$, decoding uses a *product-of-experts*:

$$p(y \mid x, E) \propto p_\theta(y \mid x) \prod_k \phi_k(y \mid x)^{\alpha_k}, \quad \alpha_k \geq 0. \tag{5}$$

Here ϕ_k are distilled from group-relative semantic advantage; weights α_k are learned by introspective calibration without gradient updates to θ.

6.3 Distributed Aggregation

Hanzo Network aggregates *priors, not gradients*. Let node priors be $\{E_i\}$. We publish hashes and quality scores; the chain computes a byzantine-robust aggregate $\tilde{E} = \text{median}\_q\{E_i\}$ under a fixed schema (token bins / embedding centroids). Conflicting contributions resolve by stake-weighted quorum plus quality caps.

7 1-Bit Semantic Compression

Inspired by BitDelta, we store only the *signs* of per-bucket deltas plus per-matrix scales. For an experience matrix $\Delta \in \mathbb{R}^{n\times m}$,

$$\widehat{\Delta} = \alpha \ \mathrm{Sign}(\Delta), \quad \alpha = \tfrac{1}{nm} \sum \_ij|\Delta\_ij|. \tag{6}$$

Scales are distilled by matching logits to a teacher rollout. We observe $\approx 29.5\times$ storage savings with negligible loss in downstream utility, enabling multi-tenant caching and rapid hot-swaps of personalizations.

8 ExperienceRegistry and P2P Sync

Registry. On-chain contract stores: content-addressed CID, Merkle root, schema version, quality vector, submitter, slashing bond. **Storage.** Off-chain IPFS/Arweave; local SQLite+LanceDB with Merkle verification. **Sync.** Gossip protocol with CRDT merge; priority given to high-quality shards (fee rebates bias peers to propagate them).

9 Token Economics ($AI)

9.1 Utility

$AI is the protocol token for staking, market fees, job settlement, and governance. *Compute credits* Θ are minted by locking $AI at current HMM rate and burned on settlement.

9.2 Emissions and Rewards

Per block, distribute R $AI: validators βR, workers γR pro-rata verified work, curators δR by experience quality shares, treasury $(1-\beta-\gamma-\delta)R$. A PoAI bonus applies: for job j with value V_j and verified cost K_j, reward ρV_j ($\rho \leq 0.1$) split among parties. Slashing burns a fraction σ of bonds on fraud.

9.3 Fees and Burns

HMM fees split to LPs and treasury; a fixed fraction ζ of market fees is burned to offset emissions. Experience submissions pay a deposit D; refunds scale with measured utility.

10 Security and Governance

Flash-& MEV-resistance: HMM quotes include dynamic risk fees; frequent batch auctions for large jobs; commitment-reveal for order flow. **Oracle bypass:** endogenous pricing limits oracle risk; optional TWAP oracles for cross-chain settlement. **Governance:** $AI holders elect parameter councils with guarded timelocks; security council can pause attesters.

11 Implementation Plan

Phase 0 (week 0–2): HMM single-pool prototype; ExperienceRegistry (Solidity); IPFS/Arweave sink. **Phase 1 (week 3–6):** Multi-asset HMM; PoAI receipts (TEE path); Zoo DSO local optimizer; GPU-accelerated retrieval (Candle tensors). **Phase 2 (week 7–12):** Verifier network;

batch auctions; DAO UI; 100+ node load test. **Phase 3 (week 13+):** ZK path pilots; security audit; mainnet.

12 Relation to Active Inference

Active inference views each agent as performing Bayesian updates; sharing beliefs resembles multiplying priors. Our TF-GRPO matches this: Eq. (5) is a product-of-experts over experiential beliefs, yielding principled, decentralized Bayesian belief propagation without weight updates.

13 Related Work

Constant-product AMMs; inventory-risk AMMs; TEEs and verifiable compute; parameter-efficient adaptation; delta compression; in-context RL; training-free alignment. (Surveyed qualitatively; implementation choices are original here.)

14 Conclusion

Hanzo Network integrates a Hamiltonian AMM for compute with decentralized, zero-training semantic learning. The result is a practical L1 for AI where market-cleared compute and pooled experiential priors compound to deliver cheaper, better, and safer AI.

A HMM Mechanics and Proof Sketches

No-arbitrage under invariant. For any feasible swap that preserves $\mathcal{H}$ net of fees, marginal price equals the gradient ratio; convex curvature and risk fees prevent cyclical arbitrage in continuous time.

Multi-asset routing. With convex $\mathcal{H}$, the path solver is a convex program; KKT multipliers interpret as SLA shadow prices.

B Solidity Interfaces (Sketch)

```
interface IExperienceRegistry {
  struct Entry {
    bytes32 merkleRoot;
    string cid; // IPFS/Arweave
    uint64  schema;
    uint64  quality; // quantized
    address submitter;
    uint256 bond;    // slashing collateral
  }
  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 get(uint256 id) external view returns (Entry memory);
}
```

```
interface IHMM {
  function quoteBuy(uint256 poolId, uint256 dTheta)
    external view returns (uint256 dPsi, uint256 fee);
  function swap(uint256 poolId, uint256 dTheta, uint256 minPsi)
    external payable returns (uint256 dPsi);
  function addLiquidity(uint256 poolId, uint256 dPsi, uint256 dTheta)
    external returns (uint256 lpShares);
}
```

C Algorithms

TF-GRPO (Training-Free) with PoE Decoding

Algorithm 1 Local ASO/TF-GRPO Step

1: **input:** query set $\mathcal{D}$, group size G, base model p_θ, current prior bank E
2: **for** $x \in \mathcal{D}$ **do**
3: Generate group rollouts $\{y^{(g)}\}\_g = 1^G$ with PoE decoding using current E
4: Score with reward model / tools to get $\{r^{(g)}\}$
5: Extract semantic advantage text A via LLM introspection over the group
6: Distill A into token/embedding buckets to produce ΔE
7: Compress ΔE to (signs, scales); append to local bank E
8: **end for**
9: Return compressed shard for registry submission

PoE Decoding

Algorithm 2 Product-of-Experts Decoding

1: logits $\boldsymbol{z} = \log p_\theta(\cdot \mid x)$
2: **for** expert k **do**
3: compute expert log-factor $\boldsymbol{h}_k = \log \phi_k(\cdot \mid x)$
4: $\boldsymbol{z} \leftarrow \boldsymbol{z} + \alpha_k \boldsymbol{h}_k$
5: **end for**
6: sample or argmax from softmax($\boldsymbol{z}$)

D Default Parameters (Initial Mainnet)

Disclaimer. This document describes a proposed protocol. Parameters and mechanisms may evolve with audit and community input.

Symbol	Meaning	Default
f_m	market fee	30 bps
f_r	risk fee coeff.	5–20 bps per % inventory move
λ	curvature	0.05
β, γ, δ	emissions split	0.35/0.50/0.10
ζ	fee burn	0.25
D	registry bond	25 $AI