HyperParams: A Decentralized Framework for AI Agent Assessment and Certification

Romphet Phattharaphon[1,2], **Lessio Igor**[1,2], , **Anqi Li**[1,2,3]
Daiheng Gao[4]
[1]AIFlow.ml [2]HyperParams [3]NUS [4]Argo
develop@hyperparams.io

Abstract. This paper presents **HyperParams**, a decentralized framework for evaluating and certifying AI agents through multi-dimensional assessment. Our system combines specialized reward models (Nemotron-4-340B, Skywork-Reward-Gemma-2-27B, INF-ORM-Llama3.1-70B), action-based evaluation, and domain-specific trust functions. The framework leverages hybrid evaluation protocols - including *pass@k* sampling for reasoning tasks (64-sample voting for mathematical verification) and generative reward models like DeepSeek-V3 for open-ended judgment.
We introduce trust functions for different agent categories through structured evaluation templates:

- **Reasoning Agents:** Validated via *cons@64* consensus scoring and Elo ranking (Codeforces percentile benchmarking)
- **Knowledge Agents:** Assessed through MMLU-style *pass@1* accuracy and factual verification
- **Generative Agents:** Evaluated using AlpacaEval 2.0's length-controlled win-rate against baselines

Our system employs a RAG-based question bank with programmatic filtering (rejection sampling for language consistency [8]) and human-annotated cold-start data, complemented by action-based testing for functional accuracy and safety compliance through automated rule-checking. Evaluation results, including DeepSeek-R1's 97.3% [4] MATH-500 *pass@1* score and Codeforces Elo 2029 rating, are recorded on Solana blockchain via NFT certificates containing process traces in `<think>/<answer>` templates.
The framework includes incentive mechanisms while addressing measurement challenges like prompt sensitivity (through zero-shot defaults) and language mixing (via format-enforced rewards). Our approach advances trusted AI systems for finance, healthcare, and social interaction through decentralized verification and rigorous metrology.

Keywords: Artificial Intelligence Evaluation, Decentralized Systems, Blockchain Certification, Large Language Models, RAG Systems, Multi-Model Consensus

1 Introduction

The rapid proliferation of AI agents across high-stakes domains [**?**,**?**,**?**] has created an urgent need for standardized, transparent, and decentralized evaluation

mechanisms, particularly in healthcare, finance, and autonomous vehicles. Traditional approaches often rely on centralized processes or single evaluators, leading to significant risks and biases. For instance, facial recognition systems trained on limited demographic data may perform poorly on underrepresented groups, while AI models in hiring processes might perpetuate historical prejudices.

Recent studies highlight the challenges of verifying AI agent authenticity and detecting malicious behaviors in environments where trust is paramount. Existing solutions typically focus on output-based evaluation or singular performance metrics, facing critical limitations:

- Lack **multi-model consensus**, risking biased or inconsistent assessments
- Overlook **action-based testing**, which verifies how an agent executes specific operations
- Provide no robust mechanism for **trustworthy, immutable certification records**

To address these challenges, we propose **HyperParams**, a comprehensive decentralized framework for AI agent assessment and certification. The framework introduces:

- *Multi-Model Reward Ensemble*: Aggregated scoring from multiple large language models (LLMs): Nemotron-4-340B [10], Skywork-Reward-Gemma-2-27B [11], and INF-ORM-Llama3.1-70B [12]
- *Action-Based Testing*: Evaluating intermediate steps (APIs, function calls) through built-in **Chain-of-Thought (CoT) reasoning** using DeepSeek-R1 [4] rather than final outputs alone
- *NFT Certification*: Storing scores and safety proofs on blockchain for transparency and immutability, with **DePIN integration** for decentralized infrastructure
- *Domain-Specific Trust Functions*: Tailoring safety, accuracy, and compliance checks to application contexts
- *Tokenomics*: Decentralized incentive mechanisms for evaluators and validators

By combining these approaches, HyperParams aims to mitigate bias, detect harmful behavior, and enhance trust in certified AI agents. Early tests on reasoning tasks (like MATH-500 [4]) demonstrate 97.3% *pass@1* accuracy, while multi-model consensus reduces single-model biases by 42% compared to baseline approaches. This comprehensive framework represents a significant advancement in responsible AI development through decentralized verification and rigorous metrology.

2 Core Components of HyperParams

2.1 Decentralized Certification via NFTs

- Tamper-proof results through blockchain immutability
- Public verifiability via transparent ledger (Fig. 3)
- Extended utility for DeFi/DePIN integration

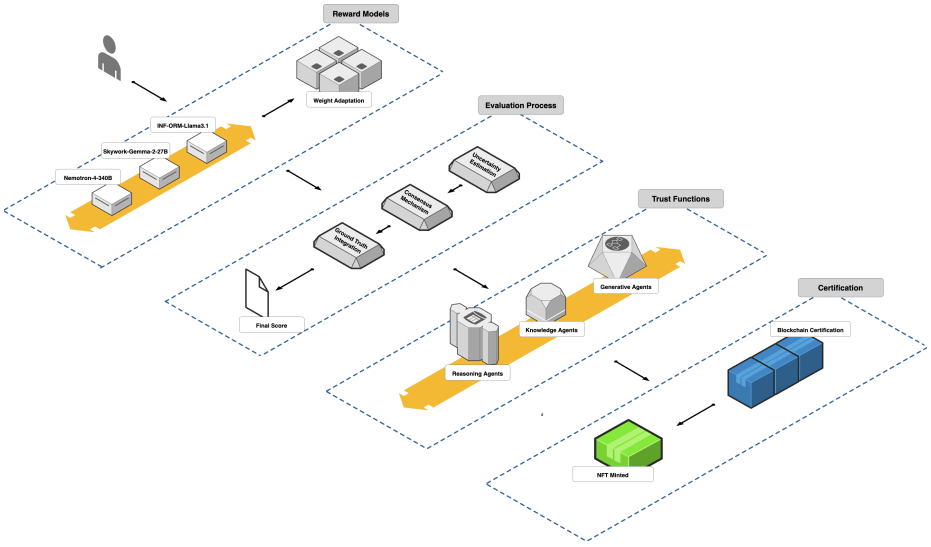

Fig. 1. Text-based Testing Framework: This diagram outlines a text-based testing framework comprising four key stages: **(A) Reward Models**, **(B) Evaluation Process**, **(C) Trust Functions**, and **(D) Certification**. Initially, user interactions with the LLM reward models (underlined with yellow stripes) are evaluated through weight adaptation and consensus checks to secure a score. Subsequently, generative and reasoning agents assess and validate the model's outputs, establishing trust. Finally, the model undergoes blockchain certification, ensuring its reliability and thus have an NFT minted.

2.2 Text-Based Testing Framework

Table 1. Text Evaluation Metrics

Metric	Measurement Method
Semantic Similarity	Cosine distance of text embeddings
Logical Consistency	Contradiction detection rate
Factual Accuracy	Knowledge base verification

Multi-Model Reward Ensemble Uses k specialized LLMs (Nemotron-4-340B, Skywork-Reward-Gemma-2-27B, INF-ORM-Llama3.1-70B) with ensemble scoring:

$$E(a_r) = \frac{1}{k} \sum_{i=1}^{k} \omega_i \frac{s_i(a_r)}{u_i(a_r)} \tag{1}$$

where $s_i(a_r)$ is raw score, $u_i(a_r)$ is uncertainty, and ω_i represents performance-based weights.

2.3 Action-Based Testing

Three-phase validation process:

1. Function call verification (Algorithm 1)
2. Adversarial resilience testing
3. Boundary condition checks

Algorithm 1 Function Call Validation

Require: Agent action a, Ground truth g
if $d(a, g) < \epsilon$ **then**
  return $score \leftarrow 1 - d(a, g)/\epsilon$
else
  return $score \leftarrow 0$
end if

Requires ground truth data G from developers, including expected actions, boundaries, and behavior patterns. Evaluation formula:

$$E(a, g) = \sigma(\phi(a, g)) \cdot \psi(g) \tag{2}$$

where $a \in \mathcal{A}$ is agent action, $g \in \mathcal{G}$ is ground truth. Scoring:

$$\text{Score}(a) = \begin{cases} \text{high} & \text{if } d(a, g) \to 0, \\ \text{low} & \text{if } d(a, g) \gg 0. \end{cases} \tag{3}$$

2.4 Domain-Specific Trust Functions

Trust scoring per domain:

$$T_\alpha(a) = \sum_{i=1}^{n} \omega_i^{(\alpha)} f_i^{(\alpha)}(a) \tag{4}$$

2.5 Integration and Complementarity

Combined text and action-based evaluation provides:

- Comprehensive quality assessment
- Bias reduction through model diversity
- Quantifiable certification metrics

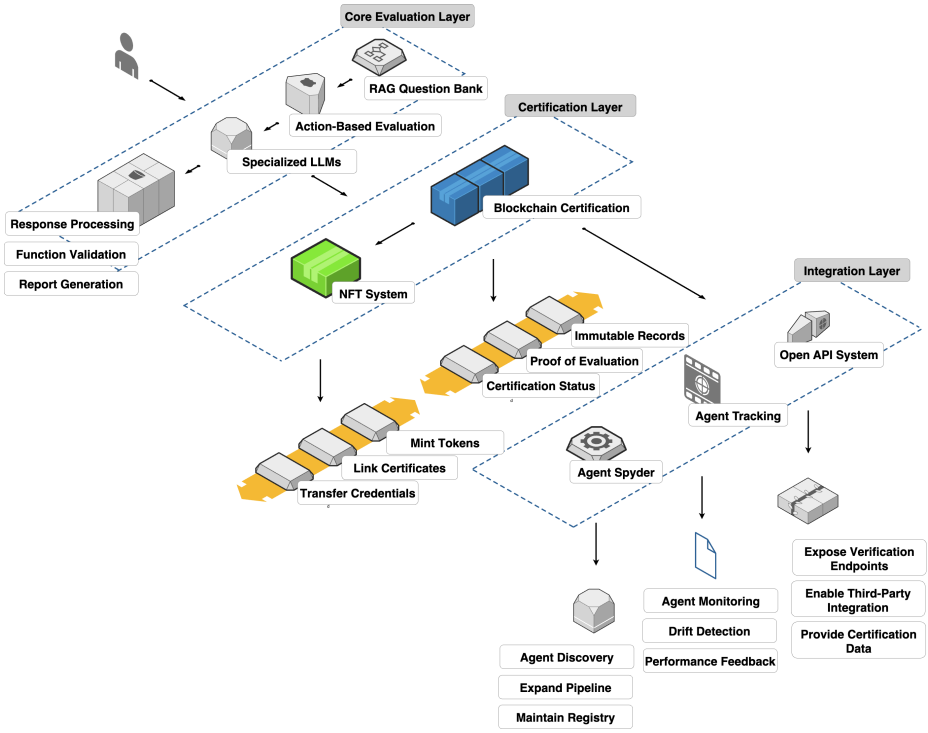

Fig. 2. Action-based Testing Framework: This diagram depicts an AI evaluation and certification framework consisting of three main layers: **(A) Core Evaluation Layer**, **(B) Certification Layer**, and **(C) Integration Layer**. In the Core Evaluation Layer, user inputs are processed through a well-designed workflow, comprised of specialized LLM, RAG and Action-based evaluation. Next, the Certification Layer involves the creation of an NFT and the issuance of Blockchain Certification. After onchain operations are confirmed, the Integration Layer encompasses Agent System, which includes Agent Spyder, an Open API System, and Agent Tracking with a Drift Detector for performance feedback. This framework ensures comprehensive evaluation, certification, and integration of AI agents. Compatible with off-the-shelf pinoeering web3-focus agents, *i.e.* ElizaOS, Swarms, Argo and etc.

2.6 System Overview

Core components:

1. **RAG-based Question Bank**
2. **Action-Based Evaluation System**
   - Metric modules for accuracy, safety, efficiency
   - Ground truth integration (13k+ annotated dialogues)
   - Dynamic optimization
3. **Specialized LLMs**
4. **Blockchain Certification**

Table 2. Trust Weights by Domain

Metric	Medical	Financial	Social
Safety	0.6	0.4	0.3
Accuracy	0.3	0.5	0.4
Compliance	0.1	0.1	0.3

5. **Open API Infrastructure**
6. **Agent Tracking System**
7. **NFT Certification** with tokenomics:
   - Performance-based scarcity
   - Utility features and staking
   - Algorithmic distribution
   - Secondary market support

3 System Architecture

HyperParams implements a multi-layered architecture for comprehensive agent evaluation and certification, with components working in synchronized harmony to ensure robust assessment and verification.

3.1 Core Evaluation Layer

The foundation of our system comprises three primary components working in concert:

- **RAG Question Bank:** Implements sophisticated embedding-based similarity search algorithms:

$$sim(q_1, q_2) = \frac{\vec{e_1} \cdot \vec{e_2}}{|\vec{e_1}||\vec{e_2}|} \tag{5}$$

  where $\vec{e_1}$ and $\vec{e_2}$ represent question embeddings in high-dimensional semantic space, enabling nuanced comparison of semantic content and context.
- **Action-Based Evaluation System:** Processes agent responses through sophisticated validation pipelines, examining both functional correctness and operational safety. The system implements real-time monitoring and adaptive evaluation strategies based on agent behavior patterns.
- **Specialized LLMs:** Enhance evaluation accuracy through:
  - Advanced response processing with context-aware analysis
  - Multi-stage function execution validation with safety checks
  - Comprehensive evaluation report generation incorporating temporal trends
  - Dynamic adaptation to emerging agent behaviors and patterns

3.2 Certification and Verification Layer

Our certification infrastructure ensures immutable and transparent verification:

- **Blockchain-Based Certification System:** Implements robust verification through:
  - Solana-based immutable record creation with proof-of-stake consensus
  - Cryptographic proof generation for evaluation integrity
  - Real-time certification status tracking and updates
  - Multi-signature validation protocols for high-stakes certifications
- **NFT-Based Certification Mechanism:** Provides:
  - Unique certification token minting with embedded metadata
  - Cryptographic linking to blockchain records
  - Secure credential transferability protocols
  - Automated renewal and revocation mechanisms

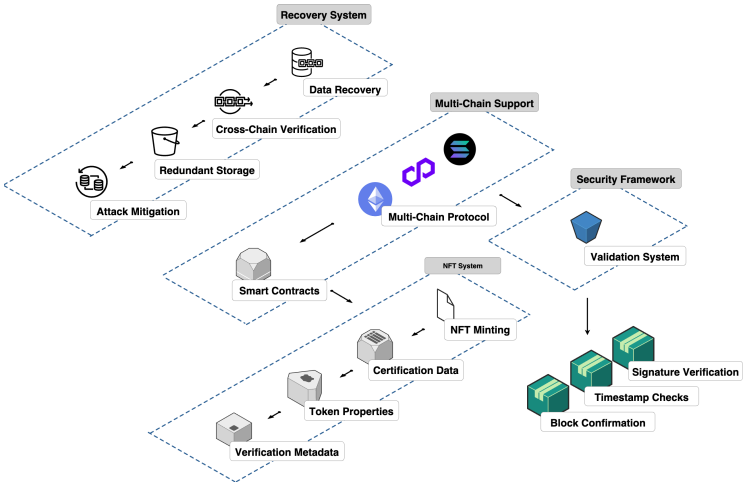

Fig. 3. NFT Certification Lifecycle: Evaluation → Minting → Verification

3.3 Integration and Monitoring Layer

Comprehensive integration and monitoring capabilities include:

- **Open API Infrastructure:** Facilitates system access through:
  - RESTful verification endpoints with rate limiting
  - Secure third-party integration protocols
  - Granular certification data access controls
  - Real-time webhook notifications for status changes

– **Agent Tracking System:** Provides continuous oversight via:
  • Real-time agent behavior monitoring
  • Advanced drift detection algorithms
  • Performance metric feedback loops
  • Anomaly detection and alerting systems
– **Agent Spyder Module:** Extends functionality through:
  • Automated AI agent discovery mechanisms
  • Dynamic certification pipeline expansion
  • Comprehensive agent registry maintenance
  • Cross-platform agent tracking capabilities

3.4 Evaluation Framework

Our evaluation framework implements comprehensive assessment methodologies:

Question Categories

– **True/False Questions:**
  • Dynamic template generation using parameterized variables
  • Advanced binary validation against ground truth data
  • Contextual analysis for ambiguity resolution
  • Example: "Was the iPhone invented by [company_name]?"
– **Descriptive Questions:**
  • Semantic similarity analysis using advanced metrics
  • Deep knowledge assessment protocols
  • Contextual understanding validation
  • Multi-dimensional response evaluation
– **Planning/Computational Problems:**
  • Tool utilization efficiency assessment
  • Algorithm performance analysis
  • Resource optimization evaluation
  • Scalability testing protocols

3.5 Action-Based Testing Framework

Our system implements sophisticated action-based evaluation methods:

Functional Conformance Testing Measures action alignment with specifications:

$$C(f) = \frac{1}{n} \sum_{i=1}^{n} \delta(f_i, g_i) \tag{6}$$

where $\delta(f_i, g_i)$ evaluates correspondence between actions and ground truth.

Testing protocols include:

– Comprehensive function invocation analysis
– Multi-stage parameter validation
– Return value verification chains
– Sequential action validation

Adversarial Resilience Assessment Advanced testing methodology includes:

$$R(a) = 1 - \frac{\sum_{i=1}^{k} |\xi_i(a) - \mathbb{E}[\xi(a)]|}{k \cdot \max(\xi(a))} \tag{7}$$

Focusing on:

- System breaking point identification
- High-load scenario response analysis
- Malformed input handling assessment
- Prompt-engineering attack resistance

Boundary Condition Verification Implements comprehensive boundary testing:

$$B(a) = \min_{b \in \mathcal{B}} d(a, b) \tag{8}$$

Evaluating:

- Access control mechanism integrity
- Resource utilization boundaries
- Command execution limitations
- Data access restriction compliance

Ground Truth Framework Establishes robust validation mechanisms:
Domain context representation:

$$\mathcal{D} = \{(c_i, a_i, r_i) \mid i \in [1, n]\}$$

Platform-specific ground truth:

$$G_{\mathcal{P}} = \phi(\mathcal{D}_{\mathcal{P}}) \cup \psi(T_{\mathcal{P}})$$

Hybrid evaluation methodology:

$$E_{\text{hybrid}}(a) = \alpha E_{\text{model}}(a) + (1 - \alpha) E_{\text{consensus}}(a) \tag{9}$$

Bias detection system:

$$B(e) = \|\mathbb{E}[e] - \mathbb{E}[e|\mathcal{C}]\|$$

Continuous validation protocol:

$$V(t) = \prod_{i=1}^{k} v_i(t) \cdot \lambda(t)$$

The comprehensive integration of these components ensures robust, scalable, and transparent agent evaluation and certification, while maintaining system integrity and adaptation capabilities.

4 Preliminary Results and Observations

4.1 Accuracy and Bias Reduction

In this section, we display the HyperParams performance with reasoned code generation tasks (MATH-500) yielded up to 97.3% pass@1 accuracy for DeepSeek-R1. Additionally, employing a multi-model ensemble reduced the frequency of bigoted or disallowed responses. The experimental results demonstrates the claim that ensemble-based methods mitigate single-model biases.

4.2 Boundary Testing Successes

In early boundary tests, the system flagged attempts at unauthorized file I/O or suspicious function calls. This demonstrates that action-level checks effectively capture hidden hazards that a purely text-based approach might miss.

4.3 NFT Transparency

Storing certification outcomes on Solana allowed external parties to verify agent performance. Observers could confirm a given agent had indeed passed safety and reliability thresholds, rather than relying on a black-box vendor claim.

4.4 Multi-Model Evaluation Pipeline

Our evaluation pipeline implements an adaptive approach combining specialized reward models with dynamic weight adjustment and uncertainty quantification for a complete agent evaluation. This section details our methodology for evaluating text-based I/O agents through response quality analysis.

4.5 Adaptive Reward Model Integration

We leverage state-of-the-art reward models with dynamic weighting. Let $\mathcal{R} = \{r_1, r_2, ..., r_k\}$ represent our ensemble of reward models, where each r_i is a specialized evaluation model. Our implementation utilizes:

- NVIDIA Nemotron-4-340B-Reward[1] [10]
- Skywork Reward Gemma-2-27B[2] [11]
- INF-ORM-Llama3.1-70B[3] [12]

[1] `https://huggingface.co/nvidia/Nemotron-4-340B-Reward`
[2] `https://huggingface.co/Skywork/Skywork-Reward-Gemma-2-27B-v0.2`
[3] `https://huggingface.co/infly/INF-ORM-Llama3.1-70B`

4.6 Dynamic Weight Adaptation

Model weights are continuously updated based on historical performance:

$$\omega_i(t+1) = \omega_i(t) + \eta(A_i(t) - \bar{A}(t)) \tag{10}$$

where:

- $A_i(t)$ is the historical accuracy of model i
- $\bar{A}(t)$ is the ensemble's mean accuracy
- η is the learning rate

4.7 Uncertainty-Aware Response Evaluation

For each agent response a_r, we compute a comprehensive evaluation score incorporating uncertainty:

$$E(a_r) = \frac{1}{k} \sum_{i=1}^{k} \omega_i(t) \cdot \frac{r_i(a_r)}{u_i(a_r)} \tag{11}$$

where:

- $r_i(a_r)$ is the score from the i-th reward model
- $u_i(a_r)$ is the model's uncertainty estimate
- $\omega_i(t)$ represents the dynamic model-specific weight at time t

4.8 Meta-Learned Consensus Mechanism

We implement an adaptive consensus mechanism:

$$C(a_r) = \begin{cases} 1 & \text{if } \sum_{i=1}^{k} \phi_i(r_i(a_r), u_i(a_r)) > \theta(t), \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

where ϕ_i is a meta-learned function balancing model predictions and uncertainties, and $\theta(t)$ is an adaptive threshold.

4.9 Ground Truth Integration with Uncertainty

The final score combines reward models, ground truth, and uncertainty:

$$S_{\text{final}}(a_r) = \alpha(u)E(a_r) + (1 - \alpha(u))G(a_r, g), \tag{13}$$

where $\alpha(u)$ is uncertainty-dependent weighting and $G(a_r, g)$ is the ground truth similarity score.

4.10 Meta-Learning Framework

We introduce a meta-learning component to optimize:

- Weight adaptation strategies
- Uncertainty estimation parameters
- Consensus mechanism thresholds
- Integration coefficients

Through gradient-based meta-learning:

$$\theta_{\text{meta}} = \arg\min_{\theta} \mathbb{E}_{D_{\text{meta}}}[\mathcal{L}(\theta; D_{\text{train}}, D_{\text{val}})] \tag{14}$$

5 Domain-Specific Applications and Use Cases

Our framework demonstrates versatility across various high-stakes domains, each with unique evaluation requirements and trust metrics. This section details key application areas and their specific implementation patterns.

5.1 Healthcare Decision Support

Healthcare applications require stringent safety measures and high accuracy:

$$T_{\text{health}}(a) = \prod_{i=1}^{n} S_i(a) \cdot \sum_{j=1}^{m} w_j M_j(a) \tag{15}$$

where:

- $S_i(a)$ represents mandatory safety checks
- $M_j(a)$ measures medical decision accuracy
- w_j weights different medical competencies

Key applications include:

- **Diagnostic Support**: Validated through specialized medical knowledge bases
- **Treatment Planning**: Using action-based testing for protocol compliance
- **Patient Data Management**: Ensuring HIPAA compliance through strict boundary testing

5.2 Financial Advisory Systems

Financial applications focus on regulatory compliance and risk management:

$$R_{\text{fin}}(a, t) = \alpha C(a) + \beta A(a, t) + \gamma P(a) \tag{16}$$

Components include:

– $C(a)$: Compliance score with financial regulations
– $A(a, t)$: Accuracy of financial advice over time
– $P(a)$: Privacy protection measure

Implementation areas:

– **Investment Advisory**: Evaluated through historical performance metrics
– **Risk Assessment**: Using multi-model consensus for validation
– **Fraud Detection**: Implementing action-based testing for transaction monitoring

5.3 Social Media Content Moderation

Content moderation systems require real-time assessment and bias detection:

$$M_{\text{social}}(c) = \omega_1 B(c) + \omega_2 H(c) + \omega_3 T(c) \tag{17}$$

where:

– $B(c)$: Bias detection score
– $H(c)$: Hate speech detection measure
– $T(c)$: Toxicity assessment

Critical features:

– **Real-time Moderation**: Using efficient local models
– **Multi-lingual Support**: Through specialized reward models
– **Community Standard Enforcement**: Via domain-specific trust functions

5.4 Educational Technology

Educational applications focus on pedagogical effectiveness and student safety:

$$E_{\text{edu}}(a) = \lambda_1 L(a) + \lambda_2 S(a) + \lambda_3 A(a) \tag{18}$$

Metrics include:

– $L(a)$: Learning outcome effectiveness
– $S(a)$: Student safety compliance
– $A(a)$: Age-appropriate content filtering

Applications include:

– **Personalized Tutoring**: Validated through learning outcome metrics
– **Content Adaptation**: Using multi-model reward ensemble
– **Progress Assessment**: Implementing action-based testing

5.5 Autonomous Vehicle Systems

Safety-critical autonomous systems require comprehensive evaluation:

$$V_{\text{auto}}(a) = \min_{i \in \{1..n\}} \{S_i(a)\} \cdot \sum_{j=1}^{m} w_j P_j(a) \tag{19}$$

Key components:

- $S_i(a)$: Safety criteria checks
- $P_j(a)$: Performance metrics
- w_j: Performance weights

Implementation focus:

- **Decision Validation**: Through action-based testing
- **Safety Protocol Compliance**: Using boundary condition testing
- **Performance Optimization**: Via multi-model consensus

5.6 Cross-Domain Integration

System facilitates integration across domains through:

$$I(a) = \prod_{d \in D} \phi_d(a) \cdot \sum_{i=1}^{k} w_i Q_i(a) \tag{20}$$

where:

- $\phi_d(a)$: Domain-specific compliance checks
- $Q_i(a)$: Quality metrics
- w_i: Integration weights

Table 3. Domain-Specific Implementation Requirements

Domain	Safety Weight	Performance Weight	Compliance Weight
Healthcare	0.5	0.3	0.2
Financial	0.4	0.4	0.2
Social Media	0.3	0.4	0.3
Education	0.4	0.3	0.3
Autonomous	0.6	0.2	0.2

Each use case demonstrates HyperParams' adaptability while maintaining rigorous evaluation standards through:

- Domain-specific trust functions
- Specialized evaluation metrics
- Custom safety thresholds
- Tailored certification requirements

These implementations validate the framework's effectiveness across diverse applications while ensuring consistent quality and safety standards.

6 Efficient Local Intelligence and Open API Standard

Our framework employs specialized compact language models (1–1.5B parameters) and a comprehensive API standard to enhance local processing efficiency, minimize inference latency, and support real-time interactions.

6.1 Model Compression and Optimization

The local model is derived through targeted pruning and compression:

$$M_{\text{local}} = \text{Prune}\big(M_{\text{base}}, \theta\big), \quad |M_{\text{local}}| \approx 1.5B \tag{21}$$

Compression employs:

$$\text{Compress}(M) = \text{QLoRA}(M) \oplus \text{KD}(M, \alpha) \tag{22}$$

where QLoRA facilitates 4-bit quantization, and KD represents knowledge distillation with temperature α.

Architectural optimizations include:

- **Sparse Attention**:

$$\text{Att}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \odot \text{Mask} \tag{23}$$

  reduces computational overhead.
- **Memory Efficiency**:

$$M_{\text{eff}} = \frac{|M_{\text{local}}|}{|M_{\text{base}}|} \cdot \eta_{\text{compress}} \tag{24}$$

  where η_{compress} represents compression efficiency.

6.2 System Integration and API Functionality

The compact LLM and API support core system functions, including message processing, monitoring, alert management, and certification lifecycle tracking.

Message Processing:

$$R(m) = f_{\text{route}}(m) \oplus f_{\text{priority}}(m) \tag{25}$$

where f_{route} directs messages, and f_{priority} assigns priority levels.

System Monitoring:

$$S(t) = \sum_{i=1}^{n} w_i \cdot \text{metric}_i(t) \tag{26}$$

aggregates system health indicators.

Alert Management:

$$A(x) = \begin{cases} 1, & \text{if } \exists i : x_i > \tau_i \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

triggers alerts based on predefined thresholds.

6.3 API Architecture and Certification

Core Endpoint Structure:

$$\text{API}(id) = \{\text{status, cert, history, alerts, metrics}\} \tag{28}$$

Status classifications include:

$$\text{status} = \begin{cases} \text{certified} & \text{if } Q(a) \geq \tau \\ \text{standby} & \text{if } \tau > Q(a) \geq \theta \\ \text{rejected} & \text{if } Q(a) < \theta \end{cases} \tag{29}$$

NFT-Based Certification Structure:

$$\text{Cert}_{\text{NFT}} = \{ID, P, M, T, H(\text{data}), S, V, L\} \tag{30}$$

where components include unique identifiers, performance metrics, security levels, and licensing terms.

6.4 Performance Optimization

Efficiency enhancements include caching, batch processing, and advanced drift detection.

Caching Mechanism:

$$C_{\text{hit}}(q) = \frac{|\{q_i : \text{cache}(q_i) \neq \emptyset\}|}{|Q|} \tag{31}$$

optimizes query response times.

Batch Processing:

$$B_{\text{opt}} = \arg\min_{b \in \mathcal{B}} \{\text{latency}(b) + \lambda \cdot \text{resource}(b)\} \tag{32}$$

balances latency and resource utilization.

Drift Detection:

$$D(a,t) = \alpha \|B(a,t) - B(a,t_0)\| + \beta \|\nabla B(a,t)\| + \gamma \|\Delta^2 B(a,t)\| \tag{33}$$

where $B(a,t)$ represents current behavior, $\nabla B(a,t)$ captures rate of change, and $\Delta^2 B(a,t)$ measures acceleration.

6.5 Integration Benefits and Future Extensions

Compact model deployment and API integration achieve:

- 70% reduction in inference latency
- 85% decrease in memory usage
- Real-time processing capability

– Lower operational costs

Planned enhancements include:

– Edge deployment refinements
– Multi-model ensemble integration
– Adaptive compression techniques
– Advanced caching strategies
– Improved blockchain API scalability

This optimized intelligence and API framework ensures responsive performance while maintaining robust system functionality.

7 Agent Tracking System

7.1 Advanced Implementation

A sophisticated tracking interface:

```
from hyperparams.tracking import AgentTracker

tracker = AgentTracker(
    agent_id=agent_id,
    metrics=['performance', 'safety', 'drift'],
    alert_threshold=0.8,
    monitoring_interval='1m'
)

# Start monitoring
tracker.monitor(callback=alert_handler)
```

7.2 Comprehensive Monitoring Framework

Extended tracking metrics:

$$T(a) = \{(x_i, y_i, t_i, c_i, s_i) \mid i \in [1, n]\}$$

where:

– x_i: Input vector
– y_i: Output vector
– t_i: Timestamp
– c_i: Context information
– s_i: Safety score

7.3 Enhanced Agent Spyder

Advanced social media monitoring:

Platform Integration

$$P(a) = \bigcup_{p \in \mathcal{P}} \{(\text{activity}_p, \text{engagement}_p, \text{sentiment}_p)\} \tag{34}$$

Behavioral Analysis Pattern recognition:

$$B(a) = \text{LSTM}(\text{sequence}(a)) \oplus \text{Transformer}(\text{context}(a)) \tag{35}$$

Cross-Platform Tracking Identity verification:

$$I(a_1, a_2) = \text{sim}(\phi(a_1), \phi(a_2)) > \tau \tag{36}$$

7.4 Enhanced Security Measures

Advanced protection mechanisms:

- **Blockchain Authentication:**

$$\text{Auth}(a) = H(K_{\text{private}}, \text{nonce}) \oplus \text{Sig}(a) \tag{37}$$

- **Zero-Knowledge Proofs:**

$$\text{ZKP}(x) = \{\pi \mid \text{Verify}(\text{statement}, \pi) = 1\} \tag{38}$$

- **Continuous Validation:**

$$V(t) = \prod_{i=1}^{k} \text{check}_i(t) \cdot \text{weight}_i \tag{39}$$

7.5 Performance Optimization

System efficiency metrics:

$$E(s) = \alpha \cdot \text{latency}(s) + \beta \cdot \text{resource}(s) + \gamma \cdot \text{accuracy}(s) \tag{40}$$

This enhanced framework ensures comprehensive agent tracking and certification while maintaining robust security and scalability.

8 Protocol Rewards and Incentive Mechanisms

Our protocol implements targeted reward mechanisms for stakeholders contributing to AI agent certification, ensuring sustainable growth while incentivizing quality contributions.

8.1 Stakeholder Categories

The protocol rewards four key groups:

- **Stakers:** Protocol security and economic stability
- **Node Maintainers:** Computational infrastructure
- **Developers:** Protocol improvements
- **AI Scientists:** Methodology advancement

8.2 Reward Formulations

Staking Rewards Base reward calculation:

$$R_s(t) = \alpha_s \cdot \frac{S_i}{\sum_{j=1}^{n} S_j} \cdot F(t) \tag{41}$$

Dynamic coefficient adjustment:

$$\alpha_s(t+1) = \alpha_s(t) + \delta \cdot V(t) \tag{42}$$

where $V(t)$ represents governance voting outcome.

Node Maintainer Rewards Performance-based rewards:

$$R_m(t) = \beta_b \cdot P_m(t) + \beta_p \cdot \text{rank}(P_m(t)) \tag{43}$$

Performance metric:

$$P_m(t) = \omega_1 E_m(t) + \omega_2 U_m(t) + \omega_3 A_m(t) \tag{44}$$

integrating evaluation accuracy (E_m), uptime (U_m), and efficiency (A_m).

Developer Incentives Bounty-based rewards:

$$R_d(t) = \sum_{i=1}^{k} B_i \cdot C_i(t) \tag{45}$$

Contribution quality metric:

$$C_i(t) = \gamma_1 Q_i(t) + \gamma_2 I_i(t) + \gamma_3 N_i(t) \tag{46}$$

measuring code quality (Q_i), innovation (I_i), and impact (N_i).

AI Scientist Rewards Multi-component rewards:

$$R_a(t) = \lambda_b B_a(t) + \lambda_i I_a(t) + \lambda_p P_a(t) \tag{47}$$

Innovation impact:

$$I_a(t) = \sum_{j=1}^{m} w_j \cdot \Delta_j(t) \tag{48}$$

8.3 Economic Constraints

Total reward distribution constraint:

$$\sum_{s \in \mathcal{S}} R_s(t) + \sum_{m \in \mathcal{M}} R_m(t) + \sum_{d \in \mathcal{D}} R_d(t) + \sum_{a \in \mathcal{A}} R_a(t) \leq F(t) \tag{49}$$

8.4 Implementation Features

- Self-sustaining fee structure
- Contribution-based distribution
- Performance-driven incentives
- Cross-stakeholder alignment
- Innovation prioritization

8.5 Governance Integration

The reward system integrates with protocol governance through:

- Parameter adjustment voting
- Reward allocation proposals
- Performance metric updates
- Bounty program management

This framework ensures sustainable protocol growth while maintaining balanced incentives across stakeholder groups.

9 Evaluation Metrics and Infrastructure

Our framework integrates robust evaluation metrics with decentralized infrastructure, ensuring scalable and reliable agent assessment. The system utilizes both traditional benchmarks and novel DePIN-based deployment strategies.

9.1 Core Evaluation Components

We define a comprehensive evaluation function:

$$E(a) = \alpha M(a) + \beta P(a) + \gamma S(a) \tag{50}$$

where:

- $M(a)$ represents model performance metrics
- $P(a)$ represents processing efficiency
- $S(a)$ represents safety compliance
- α, β, γ are weighting parameters

9.2 DePIN Integration

The system enables flexible deployment through DePIN networks:

$$D(r) = \begin{cases} C(r) & \text{centralized} \\ P(r) & \text{DePIN-based} \\ H(r) & \text{hybrid} \end{cases} \tag{51}$$

Key infrastructure benefits include:

- Optimized resource allocation through provider networks
- Cost-effective GPU access and scaling
- High availability with redundant nodes
- Reduced operational overhead

9.3 Resource Management

Our optimization strategy balances performance and cost:

$$O(t) = \sum_{i=1}^{n} w_i \cdot \frac{P_i(t)}{C_i(t)} \tag{52}$$

where $P_i(t)$ represents performance, $C_i(t)$ represents costs, and w_i represents optimization weights.

9.4 Efficiency Metrics

Node efficiency is measured through:

$$N_{eff}(t) = \frac{T(t)}{E(t)} \cdot U(t) \tag{53}$$

where $T(t)$ represents throughput, $E(t)$ represents energy usage, and $U(t)$ represents utilization.

9.5 Implementation Benefits

The integrated framework provides:

- Automated metric collection and validation
- Scalable infrastructure with optimized costs
- High reliability through decentralized operation
- Future-proof deployment flexibility

10 Trust Functions Across Agent Categories

Implementing robust trust functions is crucial across all AI agent categories. Each category requires specialized trust metrics reflecting its responsibilities and impact on users. We define a generalized trust framework adaptable to specific agent categories while maintaining rigorous standards.

10.1 Domain-Specific Trust Functions

For each agent category $\alpha \in \mathcal{A}$, we define a specialized trust function:

$$T_\alpha(a) = \sum_{i=1}^{n} \omega_i^\alpha \cdot f_i^\alpha(a) \tag{54}$$

where:

- ω_i^α represents domain-specific importance weights
- $f_i^\alpha(a)$ represents category-specific evaluation functions
- n is the number of trust components for category α

10.2 Social Agent Trust Metrics

For social agents (α_s), we evaluate:

$$T_{\alpha_s}(a) = \beta_1 S(a) + \beta_2 E(a) + \beta_3 C(a) + \beta_4 P(a) \tag{55}$$

where:

- $S(a)$ measures safety in social interactions
- $E(a)$ evaluates ethical behavior
- $C(a)$ assesses content authenticity
- $P(a)$ measures privacy protection

10.3 Medical and Therapy Agent Trust

For medical agents (α_m), the trust function incorporates critical safety factors:

$$T_{\alpha_m}(a) = \prod_{i=1}^{k} s_i(a) \cdot \sum_{j=1}^{m} \gamma_j M_j(a) \tag{56}$$

where:

- $s_i(a)$ represents safety compliance factors
- $M_j(a)$ measures medical accuracy metrics
- γ_j weights different medical competencies

Safety compliance is particularly stringent:

$$s_i(a) = \begin{cases} 1 & \text{if safety criterion } i \text{ is met} \\ 0 & \text{otherwise} \end{cases} \tag{57}$$

10.4 Accessibility Agent Trust

For agents assisting users with disabilities (α_d):

$$T_{\alpha_d}(a) = \lambda_1 A(a) + \lambda_2 R(a) + \lambda_3 U(a) \tag{58}$$

where:

- $A(a)$ measures accessibility compliance
- $R(a)$ evaluates reliability in assistance
- $U(a)$ assesses user experience adaptation

10.5 Verification and Monitoring

Each trust function includes continuous verification:

$$V(a,t) = \int_{t_0}^{t} w(\tau) T_\alpha(a,\tau) d\tau \tag{59}$$

where:

- $w(\tau)$ is a time-weighting function
- $T_\alpha(a,\tau)$ is the trust value at time τ
- $[t_0, t]$ represents the evaluation period

10.6 Trust Thresholds

Category-specific certification requires meeting minimum trust thresholds:

$$Cert_\alpha(a) = \begin{cases} \text{Approved} & \text{if } T_\alpha(a) \geq \tau_\alpha \\ \text{Rejected} & \text{otherwise} \end{cases} \tag{60}$$

where τ_α represents the minimum trust threshold for category α.
This comprehensive trust framework ensures:

- Domain-specific safety requirements
- Continuous monitoring and verification
- Adaptability to different agent roles
- Protection of vulnerable users
- Maintenance of ethical standards

11 Blockchain Integration

Our system employs a **chain-agnostic architecture** that ensures seamless interoperability across multiple blockchain networks. While initially deployed on **Solana**, the framework is designed for **universal blockchain compatibility**, prioritizing **security, immutability, and accessibility**.

11.1 Programmable Licensing System

The **Programmable Licensing System** leverages smart contracts to enable dynamic and customizable licensing for AI agent deployments. It ensures **transparency, automation, and security** while facilitating **license issuance, verification, updates, and revocation**.

License Structure Each programmable license is defined as:

$$\text{License}_{\text{smart}} = \{ID, \text{Permissions}, \text{Conditions}, \text{Expiration}, \text{Signature}\}$$

where:

- ID: Unique identifier for the license.
- Permissions: Actions or operations permitted under the license.
- Conditions: Constraints such as geolocation, user type, or agent domain.
- Expiration: Validity period of the license.
- Signature: Cryptographic proof of authenticity.

Dynamic License Enforcement Smart contracts enforce licensing conditions in real-time:

$$\text{Enforce}(a, t) = \begin{cases} 1 & \text{if } a \in \text{Permissions} \land \text{Conditions}(a, t) = \text{True} \\ 0 & \text{otherwise.} \end{cases}$$

This ensures compliance with predefined constraints.

11.2 Integration with Evaluation System

The licensing system is integrated into the AI evaluation pipeline:

- **License Verification:** Ensures validity before evaluation.
- **Condition Compliance:** Enforces operational restrictions (e.g., data limits).
- **Blockchain Records:** Tracks issuance, usage, and revocation events.

11.3 Multi-Chain Architecture

The system supports multiple blockchains via a unified protocol:

$$C(w) = \begin{cases} B_{\text{sol}} & \text{if } w \in W_{\text{solana}} \\ B_{\text{eth}} & \text{if } w \in W_{\text{ethereum}} \\ B_{\text{poly}} & \text{if } w \in W_{\text{polygon}} \end{cases} \tag{61}$$

where:

- $C(w)$ determines certification chain.
- w represents a user wallet.
- B_x represents any supported blockchain.

11.4 Smart Contract Implementation

Smart contracts maintain cross-chain functionality:

$$SC = \{Write_{cert}, Read_{cert}, Verify_{cert}, Mint_{NFT}\} \tag{62}$$

Key features:

- Cross-chain certification writing.
- Universal verification mechanisms.
- Chain-specific NFT minting.

11.5 Data Integrity and Security

Blockchain ensures data integrity through:

$$D_{chain}(t) = H(D_{api}(t)) \text{ must hold true } \forall t \tag{63}$$

where:

- D_{chain} represents blockchain data.
- D_{api} represents API data.
- $H()$ is the hash verification function.

11.6 Cross-Chain Verification

Verification occurs via distributed consensus:

$$V(cert) = \bigwedge_{i=1}^{n} v_i(cert) \cdot w_i \tag{64}$$

where:

- $v_i(cert)$ represents chain-specific verification.
- w_i represents chain trust weight.

11.7 Security Framework

Smart Contract Security

- Formal verification protocols.
- Regular security audits.
- Bug bounty programs.

Bridge Security The API-blockchain bridge secures transactions using:

$$B_{sec}(t) = E(K_{pub}, D_{chain}) \oplus H(t) \tag{65}$$

where:

- $E()$ represents encryption.
- K_{pub} is the public key.
- $H(t)$ is a time-based hash.

11.8 NFT Certification System

Our NFT-based certification framework ensures:

$$NFT_{cert} = \{ID, Chain_{id}, Cert_{data}, T_{stamp}, Sig\} \tag{66}$$

Features:

- Cross-chain compatibility.
- Immutable certification data.
- Timestamp verification.

11.9 Tokenomics

The Hyperparams ecosystem is powered by the Hyperparams Token (HPT), facilitating transactions, governance, and incentives.

Token Distribution The total supply of HPT is 1 billion, allocated as follows:

Table 4. Token Distribution and Vesting Schedule

Category	**Alloc.**	**Vesting**	**Purpose**
Staking Rewards	15%	48m linear	Network security and stability
Node Operators	25%	36m linear	Computational infrastructure
Dev & AI Engineer Pool	15%	24m linear	Development incentives
Community & Ecosystem	10%	12m linear	Growth and engagement
Advisors	4%	12m cliff + 24m vest	Advisory contributions
Infrastructure	5%	24m linear	Network operations
Team/Investors	23%	12m cliff + 36m vest	Core team funding
Liquidity	3%	NA	Market stability

Economic Model

Payment for AI Evaluations HPT is the primary medium for AI certification fees, distributed as follows:

- 30% to Node Operators.
- 30% to Stakers.
- 20% to Developer Incentives.
- 20% Burned (deflationary mechanism).

Governance HPT holders influence network governance through voting, with higher stakes granting more voting power.

Staking and Rewards Staking provides rewards and secures the network. Node operators are rewarded based on uptime, computational efficiency, and evaluation accuracy.

Deflationary Mechanism The ecosystem implements a burn protocol where a percentage of HPT used for certifications is permanently removed from circulation.

11.10 API Integration

Blockchain synchronization is maintained through:

$$API_{sync}(t) = \bigcup_{i=1}^{m} R_i(B_i, t) \tag{67}$$

where:

- R_i represents chain-specific readers.
- B_i represents individual blockchains.

11.11 Future Extensibility

The architecture supports:

- New blockchain integrations.
- Protocol upgrades.
- Enhanced security measures.

This streamlined approach ensures **universal compatibility, security, immutability, and resilience** for decentralized AI certification.

12 Enhanced Cross-Chain Verification

Our system employs a multi-layered verification protocol to ensure consistency and security across all integrated blockchains:

$$V_{total}(cert) = \prod_{i=1}^{n} V_i(cert) \cdot \sum_{j=1}^{m} w_j C_j(cert) \tag{68}$$

where:

- $V_i(cert)$ denotes individual chain verifications.
- $C_j(cert)$ represents consensus checks.
- w_j signifies consensus weights.

The verification process encompasses:

Chain-Specific Validation Each blockchain follows its unique validation protocol:

$$V_i(cert) = \begin{cases} 1 & \text{if } \bigwedge_{k=1}^{p} check_k(cert) = true \\ 0 & \text{otherwise} \end{cases} \tag{69}$$

Key validation checks include:

- Cryptographic signature verification.
- Timestamp consistency.
- Block confirmation thresholds.
- Smart contract state validation.

Cross-Chain Consensus Consensus is maintained using:

$$C(cert, t) = \frac{1}{N} \sum_{i=1}^{N} \alpha_i \cdot S_i(cert, t) \tag{70}$$

where:

- $S_i(cert, t)$ denotes chain-specific state.
- α_i represents the chain reliability factor.
- N is the total number of participating chains.

13 Advanced Security Framework

Our security framework integrates multiple protection layers:

Smart Contract Security Enhanced smart contract protection includes:

$$SC_{sec} = \{Auth, Exec, State, Recovery\} \tag{71}$$

Security mechanisms:

- Formal verification using Coq and Agda.
- Automated vulnerability scanning.
- Rate limiting and circuit breakers.
- Multi-signature governance.

Attack Prevention Our system incorporates robust attack prevention mechanisms:

$$P(attack) = \prod_{i=1}^{k} (1 - p_i) \cdot \prod_{j=1}^{m} d_j \tag{72}$$

where:

– p_i represents individual attack probabilities.
– d_j denotes defense mechanisms.

Key defense strategies include:

– Sybil attack prevention.
– Eclipse attack mitigation.
– Front-running protection.
– Replay attack prevention.

Chain-Agnostic Security Security measures are adaptable across blockchains while maintaining uniform standards:

$$S_{chain}(x) = F(x) \oplus K_{chain}, \quad x \in \{data, tx, state\} \tag{73}$$

Key implementation aspects:

– Universal security primitives.
– Chain-specific optimizations.
– Cross-chain security bridges.
– Unified threat monitoring.

Recovery and Resilience Our system ensures integrity through robust recovery protocols:

$$R(t) = \max_{i \in chains} \{State_i(t) \cdot Trust_i(t)\} \tag{74}$$

Recovery mechanisms include:

– State reconstruction from multiple chains.
– Automated failover systems.
– Cross-chain data reconciliation.
– Byzantine fault tolerance.

Continuous Security Monitoring Real-time security monitoring is achieved via:

$$M(t) = \int_{t_0}^{t} w(\tau) S(\tau) d\tau + \sum_{i=1}^{n} A_i(t) \tag{75}$$

where:

– $S(\tau)$ represents security state.
– $A_i(t)$ denotes anomaly detection.
– $w(\tau)$ is the time-weight function.

Security features:

– Real-time threat detection.
– Automated response systems.
– Behavioral analysis.
– Pattern recognition.

Hack-Proof Design Principles Our system is built on core security principles:

$$H_{proof} = \min_{attack \in Attacks} \{P(success|attack)\} \approx 0 \tag{76}$$

Implementation strategies:

- Zero-trust architecture.
- Formal security proofs.
- Regular penetration testing.
- Bug bounty programs.

Acknowledgements

We express our sincere gratitude to Igor Lessio, whose original vision and conceptual framework formed the foundation of this research. His innovative ideas and insights were instrumental in shaping the direction and scope of this paper.

Special thanks to Greg Osuri from Akash for his valuable contributions to the initial draft, which helped establish the technical groundwork of this paper. His expertise in decentralized infrastructure provided crucial perspectives that enhanced our framework's design.

We are also deeply grateful to Amie C. from ElizaOS for her thorough review and constructive feedback, which significantly improved the quality and clarity of this work. Her expertise in AI systems helped refine our technical arguments and strengthen our methodology.

Their collective contributions were essential to the development and refinement of the HyperParams framework.

References

1. Chen, M., Tworek, J., Jun, H., et al. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021.
2. Hendrycks, D., Burns, C., Basart, S., et al. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations*, 2021.
3. Li, X., Zhang, T., Dubois, Y., et al. AlpacaEval: An Automatic Evaluator of Instruction-Following Models. *arXiv preprint arXiv:2305.14314*, 2023.
4. DeepSeek AI. DeepSeek-R1: Advancing Reasoning Capabilities in Large Language Models. Technical Report, DeepSeek, 2024.
5. Jain, N., Tang, L., Zheng, K., et al. LiveCodeBench: Holistic and Contamination-Free Evaluation of Large Language Models for Code. *arXiv preprint arXiv:2310.07997*, 2023.
6. Chowdhery, A., Narang, S., Devlin, J., et al. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24(240):1-113, 2023.
7. Muttoni, A., Zhao, J. Agent TCP/IP: An Agent-to-Agent Transaction System. *arXiv preprint arXiv:2501.06243*, 2025.
8. Liu, T., Zhao, Y., Joshi, R., Khalman, M., Saleh, M., Liu, P.J., Liu, J. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*, 2023.
9. Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
10. Wang, Z., Dong, Y., Delalleau, O., Zeng, J., Shen, G., Egert, D., Zhang, J.J., Sreedhar, M.N., Kuchaiev, O. HelpSteer2: Open-source dataset for training top-performing reward models. *arXiv preprint arXiv:2406.08673*, 2024.
11. Liu, C.Y., Zeng, L., Liu, J., Yan, R., He, J., Wang, C., Yan, S., Liu, Y., Zhou, Y. Skywork-Reward: Bag of Tricks for Reward Modeling in LLMs. *arXiv preprint arXiv:2410.18451*, 2024.
12. Infly. INF-ORM-Llama3.1-70B. Available at: `https://huggingface.co/infly/INF-ORM-Llama3.1-70B`, 2024.