PolyLink: A Blockchain Based Decentralized Edge AI Platform for LLM Inference

Hongbo Liu[1], Jiannong Cao[1], Bo Yang[2], Dongbin Bai[1], Yinfeng Cao*[1], Xiaoming Shen[1], Yinan Zhang[1]
Jinwen Liang[1], Shan Jiang[3], Mingjin Zhang*[1]
[1]*Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China*
[2]*China Mobile (Hong Kong) Innovation Research Institute, Hong Kong, China*
[3]*School of Software Engineering, Sun Yat-sen University, China*
hong-bo.liu@connect.polyu.hk, csyfcao@comp.polyu.edu.hk

***Abstract*—The rapid advancement of large language models (LLMs) in recent years has revolutionized the AI landscape. However, the deployment model and usage of LLM services remain highly centralized, creating significant trust issues and costs for end users and developers. To address these issues, we propose PolyLink, a blockchain-based decentralized AI platform that decentralizes LLM development and inference. Specifically, PolyLink introduces a decentralized crowdsourcing architecture that supports single-device and cross-device model deployment and inference across heterogeneous devices at the edge. Moreover, to ensure the inference integrity, we design the TIQE protocol, which combines a lightweight cross-encoder model and an LLM-as-a-Judge for a high-accuracy inference evaluation. Lastly, we integrate a comprehensive token-based incentive model with dynamic pricing and reward mechanisms for all participants. We have deployed PolyLink and conducted an extensive real-world evaluation through geo-distributed deployment across heterogeneous devices. Results indicate that the inference and verification latency is practical. Our security analysis demonstrates that the system is resistant to model degradation attacks and validator corruptions. PolyLink is now available at https://github.com/IMCL-PolyLink/PolyLink.**

***Index Terms*—Blockchain, Web3, Edge Computing, DePIN, LLM.**

I. Introduction

Artificial Intelligence (AI) has experienced massive growth and adoption across various domains. In particular, cloud-based Large Language Models (LLMs) such as ChatGPT, Claude, and Gemini have emerged as groundbreaking AI services in recent years, demonstrating remarkable capabilities in understanding, generation, and reasoning in general tasks, thus enabling a wide range of AI applications.

However, current AI services and applications are highly centralized, with few stakeholders (e.g. cloud service providers) controlling the majority of infrastructure, models, and access. This centralization arises from several practical factors: both training and inference of large models require substantial computational and energy resources, which are typically concentrated in data centers. Normal end users, small/middle enterprises and organizations often cannot afford such hardware and software costs. Consequently, the centralized nature of current AI creates significant barriers to access and improve AI, particularly for individuals and organizations in developing regions or with limited financial resources.

AI democratization is a recent new trend aiming at addressing the AI centralization issues by redistributing AI from centralized stakeholders to decentralized parties [1]. Specifically, the democratization of AI contains three perspectives: 1) usage: making the price of AI services and applications more accessible and affordable to end users; 2) development: promoting cost-effective hardware infrastructure such as GPUs and open-source AI development framework for developers to freely develop and deploy their customized AI models (e.g., federated learning [2]); 3) governance: enabling distributed ownership and transparency management on the AI models and hardware infrastructure.

Decentralized Physical Infrastructure Networks (DePIN) are regarded as a promising solution to realize AI democratization [3]. In DePIN protocols, participants contribute their idle computational resources (e.g. GPUs and CPUs) to a shared network, which are further clustered for support AI model training and inference. Blockchain-based incentive such as tokens are used to reward the contributors to govern the network that incentivize the high-quality AI services with integrity. However, existing DePIN protocols suffer from three challenges. First, the limited computational resource provided by low-end devices makes it difficult for DePINs to support large-scale LLMs. Second, the verification mechanisms for the integrity of computational results are either insecure or inefficient, which is vulnerable to malicious device owners that try to perform dishonest computations for profit. Third, the existing DePIN incentive mechanisms is typically less effective as they only apply the similar traditional cloud service price model (device provider and users), overlooking the rewards for developers who contribute AI models.

To address these challenges, we propose PolyLink, a blockchain-based decentralized AI platform running over edge networks. We develop an inference framework that supports both single-device and cross-device execution, enabling flexible deployment across heterogeneous devices with varying model sizes and computational demands. Then we design a Trustless Inference Quality Evaluation (TIQE) protocol that ensures the inference integrity without centralized authority. At last, we develop a comprehensive incentive mechanism that

*Corresponding Author: Yinfeng Cao and Mingjin Zhang

fairly rewards device contributors and model providers based on their contributions and result quality, while charging fees to service users. We conduct extensive real-world deployment and evaluation across 20 geo-distributed devices contributed by multiple universities. Results show that the system delivers robust inference with acceptable latency across heterogeneous edge environments. The TIQE protocol achieves a favorable balance between low latency and high accuracy. Our security analysis further demonstrates resilience against model degradation and validator corruption attacks. The main contributions of this paper are as follows:

- We design POLYLINK, the first blockchain-based decentralized AI platform with full design details that supports both single-device and cross-device LLM inference over heterogeneous edge networks.
- We propose the TIQE protocol, combining lightweight Cross-Encoder model with LLM-as-a-Judge to ensure inference integrity with low cost.
- We develop a dynamic incentive model with rewards and pricing mechanisms to align the interests of model providers, workers, and validators.
- We evaluate POLYLINK through real-world large-scale geo-distributed deployment on diverse devices and provide security analysis showing resilience to model degradation and validator corruption.

II. RELATED WORK

We conduct a comprehensive review of existing trustless inference protocols and decentralized AI platforms, summarizing their key contributions and analyzing their limitations (Tab. I).

Trustless Inference Protocol. Trustless inference protocols aim to guarantee the integrity of LLM inference performed on untrusted devices. zkLLM [4] successfully provides cryptographically verifiable inference by generating a zero-knowledge proof of each result with fully decentralization. However, the method incurs substantial overhead. For LLaMa-2-13B, it requires $986\,\mathrm{s}$ to generate the model commitment and $803\,\mathrm{s}$ to generate a proof for every inference. SVIP [5] adopts an activation-based approach that trains a classifier on final-layer activations and inputs, yet it depends on a trusted third party (TTP) for training and lacks generalizability. TOPLOC [6] and SPEX [7] use locality-sensitive hashing (LSH) to hash activations and verify inference integrity, but they require re-inference by TTP and cannot efficiently distinguish high-precision model executions.

Decentralized AI platform. Recently, numerous projects have pursued decentralized AI within the Web3 ecosystem. AIArena [8] targets decentralized training: validators use public datasets to evaluate training quality. io.net [9] provides decentralized computing resources and relies on device-side status monitors and work logs to ensure that workers execute tasks honestly. SMART [10] offers a hybrid on-chain/off-chain inference framework, but its dependence on trusted execution environments (TEEs) constrains the platform's scalability and efficiency.

TABLE I: Summary of Existing Protocols and Platforms

Category	Work	Service	Decentralization	Integrity	Efficiency
Protocol	zkLLM [4]	Inference	●	Cryptography-based	○
	SVIP [5]	Inference	◐	Learning-based	◐
	TOPLOC [6]	Inference	◐	LSH-based	◐
	SPEX [7]	Computation	◐	LSH-based	◐
Platform	AIArena [8]	Training	●	Evaluation-based	◐
	io.net [9]	Computation	◐	Log-based	●
	SMART [10]	Inference	●	TEE-based	◐
	POLYLINK	Inference	●	Evaluation-based	◐

† ● = provides property; ◐ = partially provides; ○ = does not provide.

III. POLYLINK OVERVIEW

A. System Model

Fig. 1 shows the overview of POLYLINK system. There are several kinds of participants in the POLYLINK:

- **Service User:** Service users send inference requests to the system and pay with cryptocurrency tokens for computation.
- **Worker:** Participants with idle GPU resources, involving NVIDIA GPU, NVIDIA Jetson, Apple Silicon etc., contribute those resources and earn rewards. Workers are divided into two categories: *worker with single device* and *worker with multiple devices*.
- **Model Provider:** Model providers are responsible for deploying their own LLMs, such as Fine-tuning Models, on the workers and earn rewards.
- **Validator:** Validators are responsible for evaluating the inference quality responded by model on worker. To participate, validators must stake significant tokens.

B. Threat Model

POLYLINK is a decentralized platform, most system participants are untrusted, which may introduce the following threats.

Model Degradation Attack. A malicious worker may use low-precision models or perform lazy computation for profit, leading to degraded inference quality.

Validator Corruption Attack. A malicious validator may intentionally submit manipulated scores (either excessively high or low) to disrupt the consensus process, distort quality evaluation, and undermine the fairness of reward distribution.

We assume that models from providers are benign, the underlying blockchain operates correctly and remains available, and fewer than $1/3$ of validators are malicious, following the standard Byzantine Fault Tolerance assumption.

C. Design Objectives

To tackle the challenges in DePINs, the design objectives of POLYLINK are as follows.

Decentralization and Trustless. The system coordinates computation and validation among participants without relying on TTP. Both the validator committee and model deployment follow a decentralized, edge-computing paradigm, distinguishing the system from traditional cloud platforms such as OpenAI or Google. Moreover, we prioritize the devices that are actually geo-distributed.

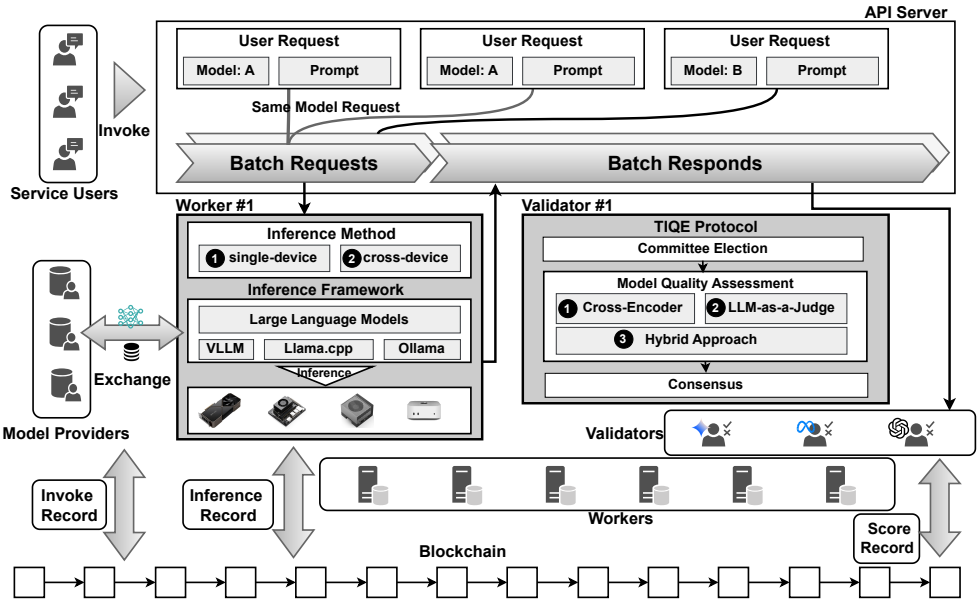


Fig. 1: Overview of PolyLink, a blockchain-based decentralized edge AI platform. Service users send inference requests via the API. The API server batches requests targeting the same model, which are then processed by a worker running that model. The responses are returned and evaluated for inference quality by the validators.

Inference Integrity. Inference requests are executed honestly on decentralized workers, and their results are evaluated by validators through a protocol to guarantee integrity.

Incentive and Protocol Efficiency. The system is designed to ensure fair and sustainable incentives for each entity. Moreover, the protocol employed by the system for integrity does not introduce substantial overhead.

IV. PolyLink Specification

A. Decentralized Model Inference

Our platform enables decentralized model inference by routing all inference requests to workers. We provide two categories of inference on workers: *single-device* and *cross-device* inference.

Single-device Inference. Model M is deployed on a single device. Typically, this approach is used for models with relatively few parameters. We define the inference process as a function

$$M(t^p) \to t^r \tag{1}$$

where a service user inputs a prompt t^p of length p into a model M. The model outputs an answer t^r of length r.

Cross-device Inference. For workers with multiple devices aim to deploy models with large parameters, we adopt the *EdgeShard* scheme [11], which partitions the model M into n sequential shards executed on an independent device:

$$M = \prod_{i=1}^{n} M_i \tag{2}$$

Due to the autoregressive nature of LLMs, inference is performed iteratively for r rounds across shards. Let $x_{0;0} := t^p$ denote the initial input. In the round $k(k \geq 1)$, for each shard M_i, the intermediate output is computed as:

$$x_{i;k} \leftarrow \begin{cases} M_i(x_{n;k-1}) & i = 1 \\ M_i(x_{i-1;k}) & i > 1 \end{cases} \tag{3}$$

After r rounds, the final response is:

$$t^r = \sum_{k=1}^{r} x_{n;k} \tag{4}$$

B. Pricing and Incentive Mechanism

1) Pricing Mechanism: For each inference request Req_i, the cost $C^i_{\text{inference}}$ is determined by the input/output token lengths p, r and the computational complexity of the model. The pricing formula is defined as: $C^i_{\text{inference}} = \delta \cdot S_M \times (C^i_{\text{in}} \times p + C^i_{\text{out}} \times r)$ where S_M denotes the normalized model scale factor, δ is a scaling coefficient that reflects market-based adjustments. C^i_{in} and C^i_{out} are the base cost per input token and output token.

2) Incentive Mechanism: In each inference batch for a selected model run on one worker, we assume a base reward per inference task is denoted as $R_i = \theta \cdot C^i_{inference}$, θ is the reward parameter. A batch contains b inference tasks, the total reward allocated to that batch is $R_{\text{batch}} = \sum_{i=1}^{b} R_i$. The reward is divided into two parts: a fixed portion $\beta \cdot R_{\text{batch}}$ allocated to validators, and a dynamic portion $(1-\beta) \cdot R_{\text{batch}}$ allocated based on the model quality score $\alpha \in [0, 1]$ (Algorithm. 1).

Worker Reward. The worker receives a share of the dynamic reward based on its model score:

$$R_{\text{worker}} = \alpha(1-\beta) \cdot R_{\text{batch}} \tag{5}$$

Validator Reward. Let there be m validators in the batch. Each validator v_i stakes an amount of token ST_{v_i}. Define the vector of staked tokens as:

$$\boldsymbol{ST} = \{ST_{v_1}, ST_{v_2}, \ldots, ST_{v_m}\}$$

The total validator reward includes both the fixed base and the remainder of the dynamic portion:

$$R_{\text{validators}} = [\beta + (1-\alpha)(1-\beta)] \cdot R_{\text{batch}} \tag{6}$$

Each validator v_i receives a share of the validator reward proportional to their staked amount:

$$R_{v_i} = \frac{ST_{v_i}}{\sum_{j=1}^{m} ST_{v_j}} \cdot R_{\text{validators}} \tag{7}$$

Algorithm 1: Reward Distribution for Inference Batch

Input: $\boldsymbol{R}$: reward per inference task vector
b: number of inference tasks in batch
α: model quality score
β: validator base reward factor
$\boldsymbol{ST}$: validator stakes
Output: R_{worker}: reward for worker
$\{R_{v_1}, \ldots, R_{v_m}\}$: rewards for validators

Procedure *RewardDistributed* $(\boldsymbol{R}, b, \alpha, \beta, \boldsymbol{ST})$**:**
  $R_{\text{batch}} := 0$
  foreach $R_i \in \boldsymbol{R}$ **do**
    $R_{\text{batch}} := R_{\text{batch}} + R_i$
  end
  $R_{\text{worker}} := \alpha \cdot (1-\beta) \cdot R_{\text{batch}}$
  $R_{\text{validators}} := [\beta + (1-\alpha) \cdot (1-\beta)] \cdot R_{\text{batch}}$
  for $j \leftarrow 1$ **to** m **do**
    $ST_{total} := ST_{total} + ST_{v_j}$
  end
  for $i \leftarrow 1$ **to** m **do**
    $R_{v_i} := \frac{ST_{v_i}}{ST_{total}} \cdot R_{\text{validators}}$
  end
End
return $R_{\text{worker}}, \{R_{v_1}, \ldots, R_{v_m}\}$

Model Provider Reward. The model provider receives rewards through transactions with the worker. Specifically, when a worker selects and utilizes a model provided by a model provider, it must pay a *model usage fee (MUF)* defined by the model provider. This fee can be fixed or dynamically priced based on model quality. The worker can also act as the model provider.

C. Trustless Inference Quality Evaluation Protocol

To ensure inference integrity and quality, we propose a *Trustless Inference Quality Evaluation (TIQE)* protocol (Fig. 2), which enables a decentralized validator committee to perform quality evaluations and reach consensus on the results. Based on the consensus, a *model quality score* is assigned to the model run on the worker.

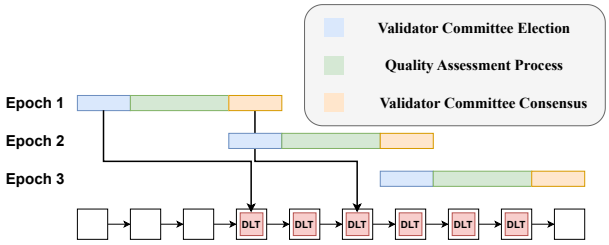


Fig. 2: Overview of the *TIQE* Protocol in each Epoch

1) Quality Assessment Process: In this process, validators evaluate inference results using a trustless scoring mechanism. The evaluation function is formally defined as:

$$\text{eval}(t^p, t^r, \mathcal{J}) \rightarrow \mathcal{J}(t^p, t^r) \rightarrow \textit{Score} \tag{8}$$

where $\mathcal{J}$ indicates the judgment approach applied to assess the result. We propose three types of judgment approaches to support efficient, cost-effective evaluation of inference quality.

Cross-encoder Approach. Cross-encoder is a light-weight Transformer-based model (e.g. BERT, LLM with few parameters) trained to measure the similarity between related query-document pairs [12], [13]. We construct an input pair by concatenating the prompt and the corresponding output result, which is then fed into a cross-encoder model to produce a quality score. This cross-encoder provides a lightweight evaluation solution that estimates the semantic similarity between the prompt and the result. However, it is limited in open generative scenario since it's limitation model capability.

LLM-as-a-Judge Approach. With the significant generalization and reasoning capabilities, LLM is widely used to evaluate the performance of other LLMs [14]. Typically, the judge LLM must be a large-scale model with strong reasoning capabilities, such as ChatGPT-o3 (175B), Gemini 2.5 Pro, DeepSeek-V2 (671B), or LLaMA 3.2 (405B). Figure 3 illustrates an example of the evaluation prompt used by the judge LLM to assess the output quality and produce a corresponding model score.

The *LLM-as-a-Judge* approach can be implemented either by invoking an commercial API or by deploying the judge model on validator nodes. However, invoking commercial APIs, such as OpenAI's GPT-o3 (which charges up to $40.0 per million tokens for output [1]), introduces significant costs. Alternatively, deploying judge models locally imposes substantial hardware requirements on validators, including high memory capacity and GPU resources.

Hybrid Approach. To mitigate the limitations of both the *cross-encoder* and *LLM-as-a-Judge* methods, we propose a hybrid evaluation approach. Within each epoch, when validators collect a batch of inference tasks associated with a specific model on a worker, they evaluate this batch using a cross-encoder. In epoch e, the cross-encoder score $Score^e_{cross}$ is defined as:

[1] http://openai.com/api/pricing/

You are an expert judge. Your task is to rate the quality of the following LLM inference result given the provided input. Rate on a scale from 1 to 5, where:
1 = Completely incorrect or nonsensical
2 = Mostly incorrect or with major flaws
3 = Partially correct but with noticeable issues
4 = Mostly correct with minor issues
5 = Completely correct, comprehensive, and well-reasoned
Input:
{input}
LLM Inference Output:
{output}
Please return only the numeric score (1 to 5) and no explanation.
Score:

Fig. 3: The prompt of *LLM-as-a-Judge* to evaluate the decentralized LLM inference quality.

$$\text{Score}^{e}_{\text{cross}} = \frac{1}{|\mathcal{B}|} \times \sum_{B \in \mathcal{B}} \sum_{i \in B} eval(t^{p}_{i}, t^{r}_{i}, \{\textit{Cross-Encoder}\}) \quad (9)$$

At a randomly selected point within the epoch, an evaluation score using the *LLM-as-a-Judge* approach, which is assigned a higher weight in the final scoring computation. In epoch e, the LLM score $Score_{llm}$ is defined as:

$$\text{Score}^{e}_{\text{llm}} = eval(t^{p}_{e}, t^{r}_{e}, \{\textit{LLM}\}) \quad (10)$$

This hybrid approach allows the system to maintain evaluation accuracy while significantly reducing computational costs, thereby providing a more cost-effective and scalable solution for trustless model quality evaluation. After a specific epoch e, the final model quality score is defined as:

$$\text{Score}^{e}_{\text{final}} = \lambda \cdot \text{Score}^{e}_{\text{llm}} + (1-\lambda) \cdot \text{Score}^{e}_{\text{cross}} \quad (11)$$

2) Model Quality Score Consensus: In the *TIQE* protocol, validators are elected at the beginning of each epoch, and they are responsible for reaching consensus on the model quality score. In this section, we introduce the design of these two key processes: validator election and consensus formation.

Validator Committee Election. At the beginning of each epoch, a validator committee is elected using a Verifiable Random Function (VRF)-based selection mechanism. When a validator $V_k \in \mathcal{V}$ observes the start of a new epoch e, it performs a VRF using its secret key sk_k and a random seed which is composed by previous block hash H_{prev} and epoch hash $\mathcal{H}(e)$.

$$(r_k, \pi_k) = \text{VRF}_{sk_k}(H_{\text{prev}} \parallel \mathcal{H}(e)) \quad (12)$$

where r_k is a verifiable pseudorandom and π_k is a proof that r_k was correctly computed with sk_k. Our system defines a public election rule $\mathcal{R}(r_k)$:

$$\mathcal{R}(r_k): \quad r_k \bmod |\mathcal{V}| < |M_{\text{committee}}| \quad (13)$$

where $|\mathcal{V}|$ is the total number of eligible validators, $M_{\text{committee}}$ is the desired committee and $|M_{\text{committee}}|$ is the committee size. If the r_k satisfies the rule, the validator becomes a member of the validator committee $M^{(e)}_{committee}$ for epoch e. The proof π_k can be used by any other node to verify that V_k was elected fairly, without revealing its secret key sk_k. This mechanism ensures decentralized, unpredictable, and verifiable election of validators in each epoch.

Validator Committee Consensus. In the consensus phase of each epoch, all validators in the elected committee submit their individual model quality scores to the smart contract SC. After SC collectes sufficient scores $\boldsymbol{Score}$, SC computes the median score $Score_{\text{med}}$ from $\boldsymbol{Score}$.

To discourage dishonest or biased evaluations, we define a deviation threshold Th. If a validator's score $Score_i \in \boldsymbol{Score}$ deviates from the median score by more than Th, i.e.,

$$|Score_i - Score_{\text{med}}| > Th \quad (14)$$

the validator is penalized by slashing a portion of staked tokens. The smart contract shown as Algorithm 2.

Algorithm 2: Scoring Consensus Smart Contract

Input: $\boldsymbol{Score}$: scores submitted by validators
$\boldsymbol{ST}$: staked tokens of validators
Th: deviation threshold
Output: $Score_{\text{med}}$: final consensus model score
$\boldsymbol{Slashed}$: slashing vector for penalized validators

Procedure *ConsensusOnScore* $(\boldsymbol{Score}, \boldsymbol{ST}, Th)$**:**
  Sort $\boldsymbol{Score}$ in ascending order
  $Score_{\text{med}} :=$ Median($\boldsymbol{Score}$)
  $\boldsymbol{Slashed} := \{0, \ldots, 0\}$ **foreach** $i \leftarrow 1$ **to** m **do**
    if $|Score_i - Score_{med}| > Th$ **then**
      $\boldsymbol{Slashed}[i] \leftarrow \gamma \cdot ST_{v_i}$
      `// γ is the slashing rate`
    end
  end
End
return $Score_{\text{med}}, \boldsymbol{Slashed}$

V. Security Analysis

In this section, we analyze the security of PolyLink under the threat model.

Theorem 1 (Model Degradation Attack Resistance): If a worker continuously performs low-precision inference, the reward obtained in the designated epoch e will be 0.

Proof 1: The worker's reward in epoch e is $R_{\text{worker}} = \alpha^{e}(1-\beta) \cdot R_{\text{batch}}$, where $\alpha^{e} \in [0,1]$. Assume the worker performs continuous low-precision inference in previous $e-1$ epochs. Then, according to the evaluation function (Eq. 11), the final quality score satisfies $\alpha_e = \lambda \cdot Score^{e}_{\text{LLM}} + (1-\lambda) \cdot Score^{e}_{\text{Cross}} \approx 0$. Thus,$R_{\text{worker}} \sim 0 \cdot (1-\beta) \cdot R_{\text{batch}} = 0$. Therefore, a worker

conducting consistently low-quality inference will receive zero reward.

Theorem 2 (Validator Corruption Attacks Resistance): If a validator continuously performs dishonestly ($< 1/3$), the staked token in the designated epoch e will be slashed to 0.

Proof 2: Let $Score_i$ be the score submitted by validator v_i in epoch e, and let $Score_{\text{med}}$ be the committee's median score. The smart contract enforces slashing if the deviation exceeds a threshold Th: $|Score_i - Score_{\text{med}}| > Th \Rightarrow \boldsymbol{Slash}[i] = \gamma \cdot ST_{v_i}, \gamma \in (0, 1]$. We consider three typical dishonest behaviors:

- **Over-scoring.** Validator gives artificially high scores to low-quality outputs. As the majority of validators are assumed honest ($< 1/3$ are malicious), the honest scores anchor $Score_{\text{med}}$ closer to the ground truth. Hence, inflated scores from v_i will exceed the threshold and trigger slashing.
- **Under-scoring.** Validator gives abnormally low scores to degrade honest workers' scores. Again, the deviation from $Score_{\text{med}}$ (which is close to honest average) leads to $|Score_i - Score_{\text{med}}| \gg Th \Rightarrow$ Slashed
- **Random-scoring.** Validator submits inconsistent or uncorrelated scores to disrupt consensus. Statistically, such behavior will periodically deviate beyond Th and accumulate slashing over epochs.

In each case, the validator's staked token is slashed by a rate γ per dishonest epoch. After k dishonest epochs, the remaining stake is: $ST_{v_i}^{(k)} = ST_{v_i}^{(0)} \cdot (1-\gamma)^k$. Taking the limit as $k \to \infty$: $\lim_{k\to\infty} ST_{v_i}^{(k)} = 0$. Therefore, any validator that persistently submits dishonest scores will eventually lose all staked tokens.

VI. Implementation and Evaluation

A. Implementation

We implement the proposed PolyLink system, integrating all previously described components. The system backend and frontend are deployed on a cloud server instance with 4-core CPU and 8G memory. We deploy the smart contracts and issue the ERC-20 token on the Sepolia testnet[2].

B. Real-world Evaluation

1) Geo-distributed Deployment: We perform a large-scale geo-distributed deployment of 20 devices from 10 different workers across multiple regions, including Hong Kong SAR, Guangzhou and Shenzhen in China and Kanazawa in Japan.

These workers operate in a decentralized edge network configuration, where each worker runs independently and participates in trustless inference tasks. The detailed hardware specifications of deployed worker nodes are shown in Table II.

2) Evaluation Metrics: We conduct evaluations from three primary aspects: *Geo-distributed Decentralized Inference Performance*, *TIQE Protocol Performance*, and *Reward Distribution*.

[2]https://sepolia.etherscan.io/address/0x9711b259e6281a1eA9465362Cb0BDd5D9Bf35AaD

TABLE II: Hardware Specifications of Geo-distributed Edge Workers

Worker	Location	Device Types	Quantity
1	Guangzhou, Panyu	NVIDIA RTX 2080 Ti	2
2	Kanazawa, Kakumamachi	NVIDIA RTX A4500	1
3	Hong Kong, Hung Hom	NVIDIA RTX 4060 Ti×1, RTX 3080 Ti×1, Jetson Orin NX 16GB×3, Jetson AGX Orin 32GB×3	8
	Shenzhen, Luohu	Apple MacBook Pro (M3 Pro)	1
4	Hong Kong, Pokfulam	NVIDIA Tesla P100	2
5	Hong Kong, Sai Kung	NVIDIA RTX 4090	8
6	Hong Kong, Hung Hom	NVIDIA RTX 3090	3
7	Hong Kong, Sha Tin	NVIDIA RTX 3090	4
8	Hong Kong, Sha Tin	NVIDIA RTX 3060	1
9	Hong Kong, Hung Hom	NVIDIA RTX 2060×1, RTX 3090×5, RTX 3090 Ti×1	7
10	Hong Kong, Hung Hom	NVIDIA Quadro GV100×2, Tesla A100×1, RTX 4090×4	7

Geo-distributed Decentralized Inference Performance. We consider several metrics: *Average Latency* and *Time to First Token (TTFT)* to measure user experience, *Output Token Throughput* and *Request Throughput* to assess system performance, and *Failure Rate* to reflect system stability.

TIQE Protocol Performance. We evaluate the TIQE protocol using two indicators:

- **Performance Overhead:** This refers to the computational and financial cost introduced by the quality assessment process. Specifically, we measure the latency of the Cross-Encoder and LLM-as-a-Judge components, as well as the cost associated with invoking LLM-as-a-Judge.
- **Degradation Detection Accuracy:** This evaluates the effectiveness of TIQE in identifying degraded outputs. We report the True Positive (TP) rate and False Positive (FP) rate for both the Cross-Encoder and LLM-as-a-Judge.

Task Reward Distribution. We consider reward distribution under different conditions to evaluate:

- **Incentive Effectiveness:** whether high-quality performers receive appropriate rewards.
- **Penalty Rationality:** whether low-quality or dishonest behaviors are properly penalized.
- **Fairness of Distribution:** whether validators receive fair rewards proportional to their stake.

C. Experimental Setup

1) Geo-distributed Decentralized Inference Performance: We deploy LLMs with different parameters on the workers: DeepSeek-R1-1.5B, DeepSeek-R1-7B, and DeepSeek-R1-14B. The models are executed using both single-device and cross-device settings. A total of 1,000 queries are sampled from the H3 dataset [15] and sent from a client located in Hong Kong.

2) TIQE Protocol Performance: We employ the Cross-Encoder model *TinyLM-L6-v2*[3] and LLM-as-a-Judge using the *DeepSeek* API[4] (0.07 $/1M input token, 1.10 $/1M output token). For degradation detection, we sample 50 true and 50 false response examples from the ShareGPT-Chinese-English-90k dataset [16], which consists of real-world dialogues between users and ChatGPT-4o.

[3]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2

[4]https://api-docs.deepseek.com/

3) Task Reward Distribution: To evaluate the proposed reward mechanism, we analyze reward distribution under varying conditions, with parameters summarized in Table III.

TABLE III: Evaluation Parameters Setting for Reward Distribution

Parameter	Description	Value
β	Portion of reward reserved for validators	0.3
θ	Scaling factor for task reward	1.0
b	Tasks per batch	32
R_i	Cost per task	$[0.1, 0.5]$
α	Score representing result quality	$\{0.2, 0.5, 0.8, 1.0\}$
$\boldsymbol{ST}$	Validator stake vectors	Case 1: [100, 100, 100] Case 2: [100, 300, 600]

D. Results and Analysis

1) Geo-distributed Decentralized Inference Performance: Table IV summarizes the inference performance. Model size significantly affects latency and throughput: models with few parameters (e.g., 1.5B) yield lower latency and higher throughput, while models with large parameters (e.g., 7B, 14B) incur substantial overhead. Geographical distance also impacts performance, workers in Hong Kong consistently achieve better results. For instance, the RTX 4060 Ti running the 1.5B model attains an average latency of $10.82\,\mathrm{s}$, TTFT of $1.02\,\mathrm{s}$, and throughput of $120.40\,\mathrm{tok/s}$; in contrast, the same model in Shenzhen shows worse performance. Cross-device inference enables execution of models with large parameters (e.g., 14B), but introduces additional overhead, with the highest latency observed at $160.05\,\mathrm{s}$. In all configurations, failure rates remain below 5%, and acceptable latency for personal use. These results confirm that PolyLink supports reliable, responsive inference across heterogeneous, geo-distributed environments.

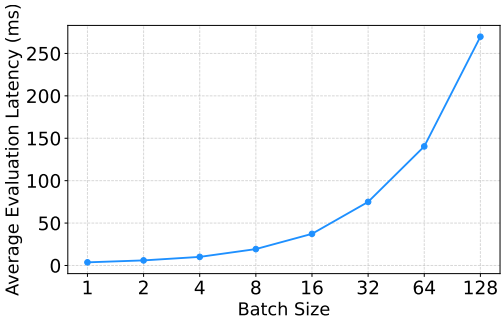
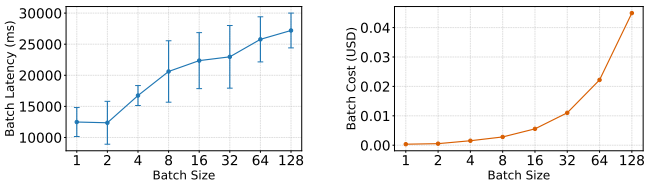


Fig. 4: Average per-batch evaluation latency of the Cross-Encoder under different batch sizes.

2) TIQE Protocol Performance: As shown in Fig. 4, the average per-batch evaluation latency of Cross-Encoder increases with batch size, from around 10 ms at batch size 1 to over 250 ms at batch size 128. This indicates that the Cross-Encoder is lightweight and scales efficiently with moderate batch sizes. Since the model is lightweight, we omit its cost in our analysis.

Fig. 5 shows the latency and cost of invoking LLM-as-a-Judge. Despite concurrent requests within each batch, latency increases with batch size due to queuing, and cost rises accordingly. This reveals a trade-off between batching efficiency and response time.



(a) Latency (b) Cost

Fig. 5: Per-Batch Latency and Cost of LLM-as-a-Judge under different batch sizes.

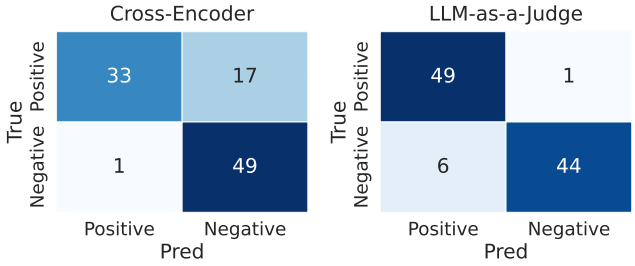


Fig. 6: Confusion matrices of cross encoder and LLM-as-a-Judge for detecting Model Degradation Attack.

Fig. 6 illustrates the confusion matrices of the Cross-Encoder and LLM-as-a-Judge on the degradation detection task. The Cross-Encoder achieves a TP rate of 66% (33/50) and a FP rate of 2% (1/50), indicating moderate detection capability with minimal false alarms. In contrast, the LLM-as-a-Judge significantly improves the TP rate to 98% (49/50), with a slightly higher FP rate of 12% (6/50).

These results show that the Cross-Encoder offers low-latency, cost-effective evaluation with acceptable accuracy, while LLM-as-a-Judge provides higher detection accuracy with increased latency and cost.

3) Task Reward Distribution: Fig. 7 shows the reward distribution results. As the model quality score increases, the worker's reward consistently rises, reflecting higher returns for higher-quality inference. In the unequal stake setting (Fig. 7b), the validator with a higher stake (Validator 3) receives the largest share among validators, demonstrating that rewards are allocated proportionally to stake. These results validate that the proposed mechanism fairly and effectively allocates rewards based on both contribution quality and economic stake.

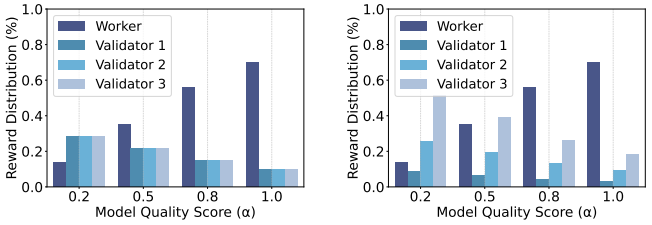
VII. Conclusion and Future Works

We proposed PolyLink, a blockchain-based decentralized AI platform that enables decentralized LLM inference across edge networks. To ensure inference integrity at low cost, we introduced the Trustless Inference Quality Evaluation (TIQE) protocol. In addition, our incentive model promotes fair and effective reward allocation among network participants. Real-world deployment and evaluation demonstrate that PolyLink

TABLE IV: Geo-distributed Decentralized Inference Performance in PolyLink.

Device Location	Device Type	Model	Latency (s)	TTFT (s)	OTPS (tok/s)	RTPS (req/s)	FR
Hong Kong	RTX4060 Ti	Deepseek-R1-1.5B	10.82	1.02	120.40	0.092	0%
	RTX3080 Ti	Deepseek-R1-7B	12.13	1.20	87.30	0.082	0%
	RTX4090	Deepseek-R1-14B	17.28	1.37	55.73	0.058	0%
Shenzhen	Apple M3Pro	Deepseek-R1-1.5B	23.30	5.65	49.46	0.041	2%
	Apple M3Pro	Deepseek-R1-7B	52.84	17.59	18.02	0.018	5%
Hong Kong	RTX3080 Ti & RTX4060 Ti	Deepseek-R1-14B	160.05	8.84	7.12	0.006	2%

† Latency: The total time from sending a request to receiving the complete response. TTFT: The time from request initiation to the first token being received. OTPS: The number of output tokens generated per second by the model. RTPS: The number of complete requests processed per second. FR: The percentage of inference requests that did not complete successfully.



(a) Equal Stake Setting. (b) Unequal Stake Setting.

Fig. 7: Reward distribution in a batch under different stake settings.

is a practical and scalable solution for decentralized AI in Web3 and DePIN ecosystems. However, the platform still has some limitations. First, the security assumption that adversaries control less than one-third of the validators may not hold in practical environments. Second, the cross-device inference method suffers from network communication latency, so the number of devices is limited. In the future, we will explore the solution to the limitations and extend the network scale of PolyLink by incorporating cross-chain support [17] and model training [18] [1]. Moreover, we plan to deploy smart city applications including digital twins and metaverse over PolyLink [19] [20].

Acknowledgment

This work is supported by CM-PolyU Joint Research Project (No. R24114H7), HK RGC Theme-based Research Scheme (No. T43-513/23-N), and Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University.

References

[1] R. Chen, Y. Dong, Y. Liu, T. Fan, D. Li, Z. Guan, J. Liu, and J. Zhou, "Flock: Robust and privacy-preserving federated learning based on practical blockchain state channels," in *Proceedings of the ACM on Web Conference 2025*, ser. WWW '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 884–895. [Online]. Available: https://doi.org/10.1145/3696410.3714666

[2] M. Cao, L. Zhang, and B. Cao, "Toward on-device federated learning: A direct acyclic graph-based blockchain approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 4, pp. 2028–2042, 2021.

[3] Z. Lin, T. Wang, L. Shi, S. Zhang, and B. Cao, "Decentralized physical infrastructure networks (depin): Challenges and opportunities," *IEEE Network*, 2024.

[4] H. Sun, J. Li, and H. Zhang, "zkllm: Zero knowledge proofs for large language models," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 4405–4419.

[5] Y. Sun, Y. Li, Y. Zhang, Y. Jin, and H. Zhang, "Svip: Towards verifiable inference of open-source large language models," *arXiv preprint arXiv:2410.22307*, 2024.

[6] J. M. Ong, M. Di Ferrante, A. Pazdera, R. Garner, S. Jaghouar, M. Basra, and J. Hagemann, "Toploc: A locality sensitive hashing scheme for trustless verifiable inference," *arXiv preprint arXiv:2501.16007*, 2025.

[7] M. Dallachiesa, A. Pitasi, D. Pinger, J. Goodbody, and L. Vaello, "Statistical proof of execution (spex)," *arXiv preprint arXiv:2503.18899*, 2025.

[8] Z. Wang, R. Sun, E. Lui, T. Zhou, Y. Wen, and J. Sun, "Aiarena: A blockchain-based decentralized ai training platform," in *Companion Proceedings of the ACM on Web Conference 2025*, 2025, pp. 1375–1379.

[9] IO.net, "IO.net: Decentralized gpu cloud," https://io.net/, 2025, accessed: 12 Jun 2025.

[10] J. Huang, L. Kong, G. Cheng, Q. Xiang, G. Chen, G. Huang, and X. Liu, "Advancing web 3.0: Making smart contracts smarter on blockchain," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 1549–1560.

[11] M. Zhang, X. Shen, J. Cao, Z. Cui, and S. Jiang, "Edgeshard: Efficient llm inference via collaborative edge computing," *IEEE Internet of Things Journal*, 2024.

[12] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[13] Z. Zhang, Y. Rao, H. Xiao, X. Xiao, and Y. Yang, "Proof of quality: A costless paradigm for trustless generative ai model inference on blockchains," *arXiv preprint arXiv:2405.17934*, 2024.

[14] J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu *et al.*, "A survey on llm-as-a-judge," *arXiv preprint arXiv:2411.15594*, 2024.

[15] B. Guo, X. Zhang, Z. Wang, M. Jiang, J. Nie, Y. Ding, J. Yue, and Y. Wu, "How close is chatgpt to human experts? comparison corpus, evaluation, and detection," *arXiv preprint arxiv:2301.07597*, 2023.

[16] shareAI, "Sharegpt-chinese-english-90k bilingual human-machine qa dataset," https://huggingface.co/datasets/shareAI/ShareGPT-Chinese-English-90k, 2023.

[17] Y. Cao, J. Cao, D. Bai, L. Wen, Y. Liu, and R. Li, "Map the blockchain world: A trustless and scalable blockchain interoperability protocol for cross-chain applications," in *Proceedings of the ACM on Web Conference 2025*, 2025, pp. 717–726.

[18] Z. Wang, R. Sun, E. Lui, T. Zhou, Y. Wen, and J. Sun, "Aiarena: A blockchain-based decentralized ai training platform," in *Companion Proceedings of the ACM on Web Conference 2025*, ser. WWW '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 1375–1379. [Online]. Available: https://doi.org/10.1145/3701716.3715484

[19] Y. Cao, J. Cao, D. Bai, Z. Hu, K. Wang, and M. Zhang, "Polyverse: An edge computing-empowered metaverse with physical-to-virtual projection," in *2023 International Conference on Intelligent Metaverse Technologies & Applications (iMETA)*, 2023, pp. 1–8.

[20] Y. Cao, J. Cao, B. Du, and R. Li, "Decentralized digital twin networks," *IEEE Communications Magazine*, pp. 1–7, 2025.