CrowdBA: A Low-Cost Quality-Driven Crowdsourcing Archi-tecture for Bounding Box Annotation Based on Blockchain

Shenglong Liao , Rongxin Guo * , Jianqing Zhu

Posted Date: 28 November 2024

doi: 10.20944/preprints202411.2183.v1

Keywords: Ethereum Blockchain; Smart Contract; Crowdsourcing; Data Annotation; Bounding Box Fusion Algorithm

Article

CrowdBA: A Low-Cost Quality-Driven Crowdsourcing Architecture for Bounding Box Annotation Based on Blockchain

Shenglong Liao, Rongxin Guo * and Jianqing Zhu

School of Engineering, Huaqiao University, Quanzhou 362000, China
* Correspondence: 42464393@qq.com

Abstract: Many blockchain-based crowdsourcing frameworks currently struggle to address the high costs associated with on-chain storage and computation effectively, and they lack a quality-driven incentive mechanism tailored to bounding box annotation scenarios. To address these challenges, this paper proposes CrowdBA: a low-cost quality-driven crowdsourcing architecture. The CrowdBA utilizes the Ethereum public blockchain as the foundational architecture and develops corresponding smart contracts. First, by integrating Ethereum with the InterPlanetary File System (IPFS), storage and computation processes are shifted off-chain, effectively addressing the high costs associated with data storage and computation on public blockchains. Additionally, the CrowdBA introduces a Dynamic Intersection over Union Weighted Bounding Box Fusion (DWBF) algorithm, which assigns dynamic weights based on IoU to infer true bounding boxes, thereby assessing each worker's annotation quality. Annotation quality then serves as a key criterion for incentive distribution, ensuring fair and appropriate compensation for all contributors. Experimental results demonstrate that the operational costs of each smart contract function remain within reasonable limits; the off-chain storage and computation approach significantly reduces storage and computation expenses, and the DWBF algorithm shows marked improvements in accuracy and robustness over other bounding box fusion methods.

Keywords: ethereum blockchain; smart contract; crowdsourcing; bounding box annotation; bounding box fusion algorithm

1. Introduction

1.1. Background

Bounding box annotation is crucial for training models in object detection, image recognition, and machine learning, as it provides essential ground-truth data that enhances the model's ability to learn and interpret visual information efficiently [1,2]. With the rapid advancements in deep learning and artificial intelligence, there is an increasing demand for vast amounts of high-quality annotated data [3,4]. In practice, businesses and research institutions often outsource data annotation tasks to centralized data labeling companies, which then hire workers to perform the annotations. However, this centralized approach presents several challenges: First, centralized data annotation companies are responsible for handling and storing large volumes of sensitive information. Should vulnerabilities exist in the data management or storage systems, this can lead to unauthorized access, data theft, or leakage, thereby increasing the risk of data breaches [5,6]. Additionally, incentive distribution is often controlled by the company, making it difficult to ensure fair compensation for workers, which can dampen their motivation. Blockchain technology, with its decentralized, transparent, and immutable characteristics, has been widely applied in fields such as Information traceability [7], IoMT [8], and the Supply Chain [9]. When applied to crowdsourcing systems, blockchain offers distinct advantages in data security, transparent incentive distribution, and trust

 doi:10.20944/preprints202411.2183.v1

management. By employing decentralized storage and data encryption, blockchain reduces the security risks associated with centralized data storage, mitigating the potential for insider-driven data leaks [10]. Furthermore, blockchain-based smart contracts automate and transparently manage incentive distribution, ensuring that rewards are allocated fairly and according to predefined rules, thereby enforcing contract terms accurately and effectively [11]. Finally, through distributed consensus, blockchain establishes a decentralized trust foundation that enables efficient collaboration between task providers and participants without the need for mutual trust, significantly enhancing the system's reliability and user engagement [12].

1.2. Related Work

Extensive research has been conducted on blockchain-based crowdsourcing architectures in recent years. Shu Yang proposed a Web 3.0-based data annotation service platform that uses blockchain technology to securely store user information and transaction records, while applying game theory to optimize pricing strategies and maximize utility for all parties. However, due to the high cost of smart contract execution, the platform employs a consortium blockchain architecture, limiting its decentralization and security. Additionally, the platform lacks a quality-driven incentive mechanism, which can hinder the assurance of high-quality annotation services for requesters [13]. Witowski et al. introduced a platform named MarkIt for collaborative annotation of medical imaging data, which incorporates blockchain technology. Its token-based incentive distribution system improves participation and annotation efficiency but lacks the ability to assess annotation accuracy, restricting its applicability to collaborations among medical experts or teams and limiting its scalability to broader public crowdsourcing tasks [14]. The research team led by Zhu Xiangrong developed FactChain, a platform designed for cross-organizational knowledge sharing and fusion. It uses a truth inference algorithm to assess the credibility and quality of shared knowledge, thus providing quality-driven incentives to contributors. However, as FactChain operates on a consortium blockchain, there is room for improvement in its security [15]. Li M et al. developed CrowdBC, a platform built on the Ethereum public blockchain to support decentralized task management and verification. It employs a reputation management system and time-locked deposit mechanism to incentivize contributions. Nevertheless, CrowdBC relies on manual review to verify submission quality, which incurs high costs and fails to protect workers from potential exploitation by requesters who might undervalue their contributions [16].

Building on the principles outlined above, this paper aims to develop CrowdBA, a low-cost, quality-driven crowdsourcing architecture for bounding box annotation. The main contributions of this study are as follows:

- Design a CrowdBA architecture for bounding box annotation using the Ethereum public blockchain as the underlying framework. By developing corresponding smart contracts and integrating the Ethereum public chain with the InterPlanetary File System (IPFS), we shift storage and computation off-chain, reducing the high costs associated with data storage and computation on the blockchain and improving scalability;
- Develop a DWBF algorithm based on Intersection over Union (IoU) for dynamically weighted bounding box fusion. This algorithm infers a true bounding box based on the annotations provided by workers, allowing for the evaluation of each worker's annotation quality. Annotation quality serves as a crucial factor for incentive allocation, ensuring fair reward distribution based on the accuracy of each worker's contributions;
- Deploy the developed smart contracts on the Ethereum test network to assess the cost-effectiveness of each functional contract, validating the protocol's practical feasibility. We also evaluate the accuracy of the DWBF algorithm, demonstrating its effectiveness in producing high-quality annotations.

This paper is divided into six chapters. The structure is as follows: Section 2 reviews the relevant concepts of blockchain and smart contracts. Section 3 details the design of CrowdBA architecture. Section 4 outlines the implementation steps of the DWBF algorithm. Section 5 presents performance

 doi:10.20944/preprints202411.2183.v1

and accuracy testing of the smart contracts and DWBF algorithm. Finally, Section 6 concludes the paper and discusses future research directions.

2. Preliminary

2.1. Blockchain and Smart Contracts

Blockchain is a decentralized, distributed ledger technology, structured as a chain of blocks linked sequentially through cryptographic hash functions. Each block connects to its predecessor, forming an immutable chain that guarantees data security and transparency [17,18]. As shown in Figure 1, each block consists of a block header and a block body. The block header includes the hash value of the previous block (parent hash), which ensures continuity and integrity across the blockchain. The block body stores transaction data in a Merkle tree structure, generating a Merkle root to enable efficient verification of transaction integrity within the block [19]. This structure upholds blockchain immutability: if any data in a block is tampered with, all subsequent blocks would require recalculation and validation, significantly raising the cost of malicious modifications and thereby enhancing blockchain security.Smart contracts are auto-mated protocols implemented on the blockchain, allowing contract terms to be executed automatically according to predefined rules and conditions. This automation, combined with blockchain's decentralized architecture, eliminates the need for intermediaries and ensures the transparency and efficiency of contract execution, enabling multi-party collaboration in a trustless environment [20,21].

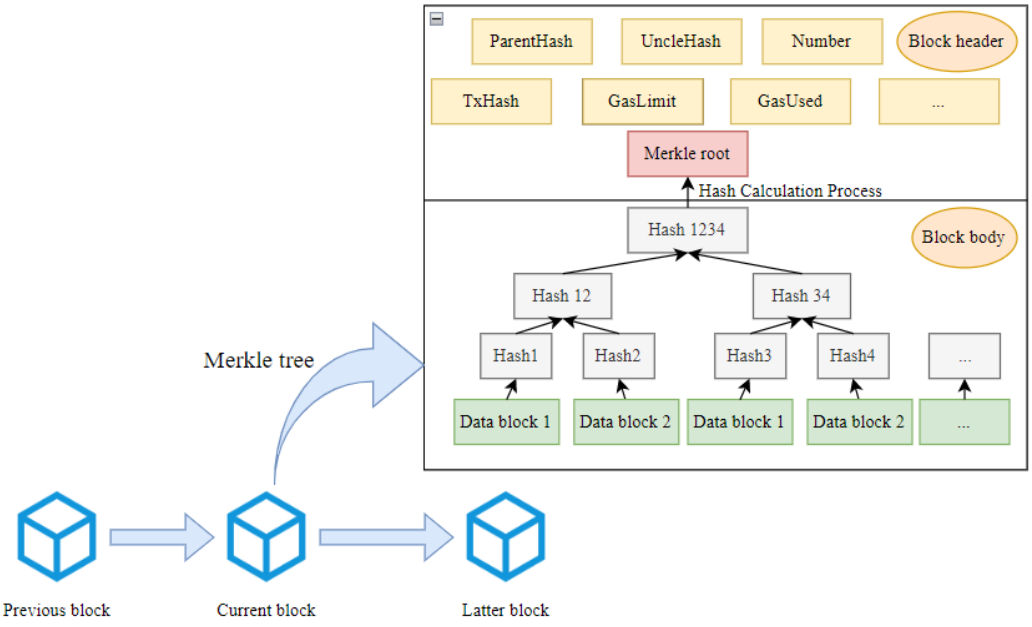

Figure 1. Block Structure.

2.2. Types of Blockchain

Blockchain can be categorized into three main types—public, consortium, and private blockchains—based on differences in access permissions, consensus mechanisms, and data transparency [22–25]. Table 1 provides a systematic comparison of these blockchain types. Public blockchains offer the highest degree of decentralization and security, facilitating transparent and decentralized data management in environments where trust among participants is absent. Consortium blockchains, managed by specific organizations or groups, reduce decentralization and are maintained by pre-approved nodes. Due to the limited number of nodes, consortium blockchains achieve higher performance than public blockchains, although they require some level of trust between nodes, making them suitable for collaborative scenarios involving multiple organizations. Private blockchains are designed for single-entity applications, with one organization fully controlling the network. They offer the best processing speed and transaction throughput but the

 doi:10.20944/preprints202411.2183.v1

least transparency, as data remains internal, posing a risk of internal abuse. Given the crowdsourcing activities inherently rely on widespread public participation, this architecture utilizes the Ethereum public blockchain as the underlying architecture. Through a well-designed smart contract, this architecture achieves high decentralization, security, and transparency while managing gas fees within user-acceptable limits.

Table 1. Blockchain Types.

	Public Blockchain	Consortium Blockchain	Private Blockchain
Participants	Anyone	Members	Internal access only
Consensus Mechanism	PoW / PoS / DPoS	RAFT / PBFT / IBFT	RAFT / PBFT / IBFT
Bookkeepers	All nodes	Consortium members	Individual
Incentive Mechanism	Required	Optional	N/A
Degree of Decentralization	Fully decentralized	Multi-centralized	Centralized
Security	High	Medium	Low
Cost	High	Medium	Low
Transparency	Public	Consortium-only	Internal only
Mbps	20-100 tx/s	1,000-10,000 tx/s	10,000-100,000 tx/s
Typical Use Cases	Crypto, DeFi, NFTs	Cross-org payments, SCM	Internal audit, data mgmt

3. CrowdBA Architecture

3.1. Architecture Design

The proposed CrowdBA architecture is illustrated in Figure 2. The CrowdBA uses Ethereum as its foundational framework and employs crowdsourcing as the method for bounding box annotation, the core functionality of the system. Key roles include the requester, worker, storage server, and smart contract. The roles and primary functions within this crowdsourcing architecture are detailed below:

1. Requester: Defined as the initiator of crowdsourcing tasks for bounding box annotation on the blockchain. Any user on the blockchain can post crowdsourcing tasks through CrowdBA, becoming a requester.
2. Worker: Defined as the executor of crowdsourcing tasks for bounding box annotation on the blockchain. Users can claim appropriate tasks within the CrowdBA to perform annotations, receiving incentives provided by the requester in return.
3. Storage Server: Composed of Ethereum and IPFS (InterPlanetary File System). IPFS is a distributed protocol for file storage, sharing, and retrieval. The system generates a content hash (CID, Content Identifier) for each data item through hash computation, which uniquely identifies the data within the IPFS network. Annotation tasks and results are stored and exchanged via IPFS, alleviating storage burdens on the blockchain.
4. Smart Contract: The smart contract plays a pivotal role in ensuring algorithm transparency and automating incentive distribution. It is responsible for recording and verifying data, executing predefined incentive distribution rules, and managing the posting and acceptance of tasks.

Requesters create smart contracts containing specific task terms and deploy them on the Ethereum blockchain. Bounding box annotation workers interact with the smart contract to apply for and complete tasks. Ultimately, the smart contract evaluates each worker's contribution to the task and distributes incentives accordingly.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 November 2024 doi:10.20944/preprints202411.2183.v1

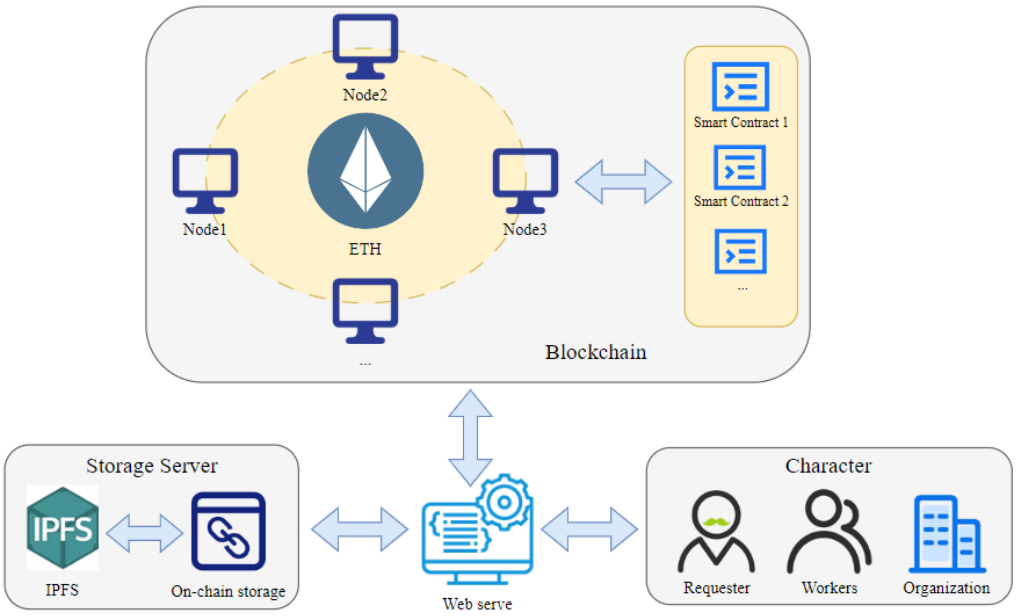

Figure 2. CrowdBA architecture.

In this paper, we assume that a requester D_i has published a contract instance C via a contract factory, which consists of a set of tasks $T = \{\tau_1, \tau_2, \ldots, \tau_m\}$, where τ_i represents the i-th bounding box annotation task. Suppose there are K workers, represented as $W = \{\varrho_1, \varrho_2, \ldots, \varrho_k\}$, and worker ϱ_i is interested in a subset $T_i \in T$ of tasks and applies to participate. Requester D assigns a subset of annotation tasks $S = \{\tau'_1, \tau'_2, \ldots, \tau'_m\} \in T_i$ to each worker. Each worker W_i is assigned a weight representing their quality contribution. The requester D defines an IPFS node for distributed storage of both the tasks T and the data results generated by workers W.

Let ψ denote the minimum dynamic weight threshold for worker participation in the DWBF (Dynamic Weighted Bounding Box Fusion) algorithm. If the Intersection over Union (IoU) weight of the task τ_i submitted by worker ϱ_i is below ψ, their annotation on data τ_m fails to meet the quality requirement. The ground truth bounding box τ_m^* for data τ_m is computed based on the weights ω_k of each worker, followed by the calculation of each worker's reward R_i, which is then distributed by the smart contract as an incentive. The symbols and their definitions used in CrowdBA are listed in Table 2.

Table 2. Symbols and Definitions.

No.	Symbol	Definition
1	T, T_{itask}	Task set, eligibility verification task group
2	τ, τ_m^*	Bounding box annotation task, ground truth of bounding box
3	D	Requester
4	W, ϱ	Worker set, eligibility worker
5	C	Contract instance
6	Ψ	Minimum weight threshold
7	R	Worker reward
8	S_T, S_H	Contract parameters, task parameters
9	t_{end1}, t_{end2},M	Data submission deadline, reward calculation deadline, maximum number of workers
10	Desc_T	Task description
11	Ad_T, Ad_R, Ad_W	Task address, requester address, worker address
12	H_d	Data hash

 doi:10.20944/preprints202411.2183.v1

3.2. Design of Smart Contract

The CrowdBA architecture implements its functionality through two smart contracts, each encompassing core functions: the Factory Contract and the Task Contract. This section focuses on the seven key processes of the smart contracts within the CrowdBA architecture, detailing the functions and core operations at each stage. The stages and core functions are illustrated in Figure 3.

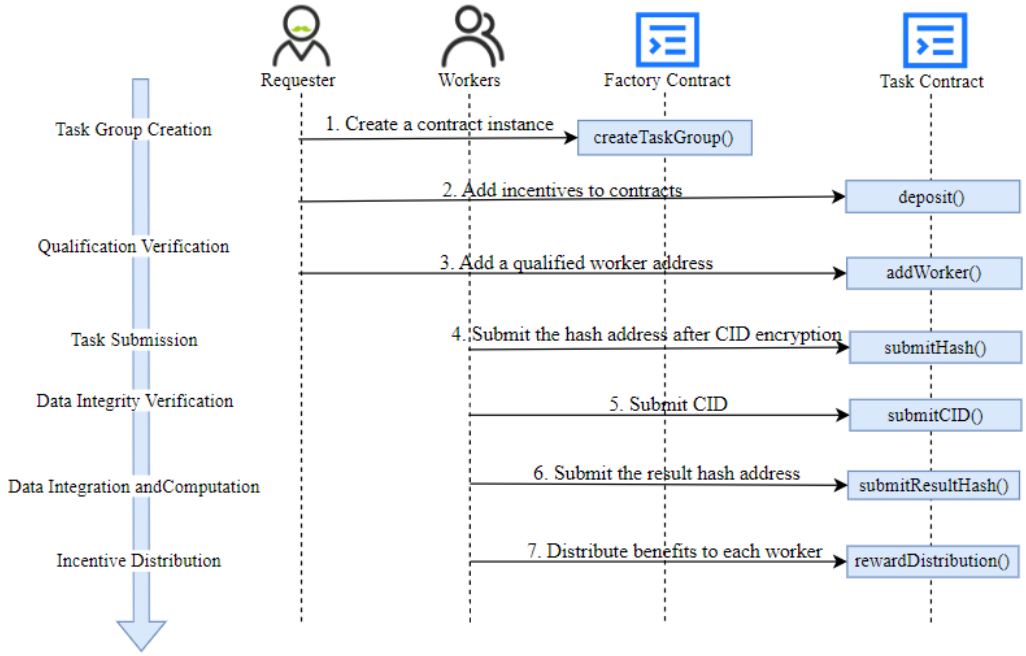

Figure 3. Smart Contract Workflow Diagram.

1. Task Group Creation Stage: The requester D_i calls the task group creation function, entering contract parameters S_T through the factory contract to dynamically create and manage each task contract instance C. Task parameters S_H are stored by the requester on IPFS. By encapsulating task group creation logic within the factory contract, the smart contract enables modular task management and data encapsulation. Each task group T_i exists as a distinct contract instance C_i, simplifying management and interaction. A payable staking function is defined within the contract to allow the requester D_i to add incentives R at any time; without sufficient R, the contract may fail to attract workers W. The contract and task parameters are defined as follows:

$$\begin{cases} S_T = \{\text{tend}_1, \text{tend}_2, M\} \\ S_H = \{\text{Desc}_\text{T}, \text{T}_{itask}, \psi, \text{R}\} \end{cases} \quad (1)$$

2. Eligibility Verification Stage: During eligibility verification, worker ϱ_i reviews task parameters S_H to examine the requester-defined task group T_i and the eligibility verification tasks T_{itask}. This task set includes image data, annotation requirements, and a corresponding one-dimensional array $< Img_{ID} >$. Worker ϱ_i assesses factors such as task difficulty and annotation cost to decide whether to accept the task. Upon acceptance, the worker extends τ_{itask} off-chain into a list of six-element arrays in the format $< Img_{ID}, label, x_{min}, y_{min}, x_{max}, y_{max} >$, compiling the annotation results T_{itask} and submitting them for requester review. After confirming the worker's qualification, the requester calls the add-worker function to add the worker's address Ad_W to the eligible workers mapping. This function can only be invoked by the contract creator D_i and only workers recorded in this mapping can call the data submission function to proceed with annotation tasks. As shown in

 doi:10.20944/preprints202411.2183.v1

Figure 4, eligibleWorkers is a mapping from addresses to Boolean values, storing the addresses of workers qualified to submit data. The eligibility verification function first checks that the caller is the contract deployer and that the number of eligible workers is below the maximum limit. Only when these conditions are met can the Function 1 be executed. Upon a successful call, the provided worker address Ad_W is set to true, and the current worker count increments by one.

Function 1: AddWorker

input: workerAddress
output: workerAdded ← true

1. **if (**msg.sender == owner) **then**
2. **if (**currentEligibleWorkers < maxWorkers) **then**
3. [eligibleWorkers] ← workerAddress
   currentEligibleWorkers + 1
   workerAdded ← true;
4. **else**
5. Return error;
6. **else**
7. Return error;
8. **End**

3. Task Submission Stage: When the first eligible worker ϱ_1 with address Ad_W is added to the eligible workers mapping, the contract enters the task submission stage. The requester D_i uses an off-chain client to send a set of images and the corresponding one-dimensional annotation array $< Img_{ID} >$ for task T_i to ϱ_1 for annotation. After completing the annotation for task T_i, ϱ_1 submits the annotation results in a six-element array format $< Img_{ID}, label, x_{min}, y_{min}, x_{max}, y_{max} >$ denoted as $T_{iresult}$, to the IPFS system, which returns a Content Identifier (CID). The CID serves as both the address and root hash of the Merkle Directed Acyclic Graph (DAG) representing the content in IPFS. The CID generation process is shown in Equation (2): data is divided into 256KB blocks, $Data = \{B_1, B_2, \dots, B_n\}$. Each data block B_i is hashed to produce a unique identifier $h_i = H(B_i)$. These hash values are then recursively hashed to construct a Merkle tree until reaching the root node, where the root hash (the topmost unique node) serves as the unique identifier for the entire dataset. The hash value $h_j^{(k+1)}$ at the highest level $k+1$ is the CID.

$$\begin{cases} h_i^{(k)} = H(B_i), & k = 0 \\ h_j^{(k+1)} = H\left(h_{2j-1}^{(k)} \parallel h_{2j}^{(k)}\right), & k > 0 \end{cases} \tag{2}$$

4. Data Integrity Verification Stage: The smart contract enters the data integrity verification stage when either the time reaches t_{end1} or the number of workers who have submitted hash values H_d reaches K. In this stage, workers are required to call the Function 2, uploading the CID generated during the task submission stage to the smart contract for verifying the actual IPFS content. The smart contract then performs a hash calculation on the received CID to ensure data integrity and authenticity. The data integrity verification process is illustrated in Figure 5. In the CID submission function, the smart contract first checks that the caller has not previously submitted a CID and ensures that the function can only be invoked when the current time has reached t_{end1} or the number of workers who have submitted data meets the threshold K.

 doi:10.20944/preprints202411.2183.v1

Within the function, the Keccak-256 hash function is called to convert the worker's CID into a 256-bit byte hash value H_d. This hash value is then compared to the hash submitted by the caller during the task submission stage. If the values match, the CID is stored in the submitted array, and a CID submission event is triggered. Finally, the function returns true, indicating successful submission; otherwise, it returns false, indicating a failed submission.

Function 2: CIDSubmission

Input: CID
Output: CIDSubmitted ← ture

If (currentTime ≥ timeEnd1) **then**
  If (maxWorkers ≥ submittedWorkers) **then**
    If (WorkerSubmissions[msg.sender] = NULL) **then**
      If (Keccak-256(CID) = H_d) **then**
        [CIDSubmission] ← CID
        submittedWorkers + 1
            CIDSubmitted ← ture;
      else
        Return error;
    else
      Return error;
  else
    Return error;
else
  Return error;

5. Data Integration and Computation Stage: After confirming that the hash value H_d submitted by worker ϱ_i matches the previously submitted IPFS address CID, the hash H_d is added to the array of successfully submitted CIDs. Workers ϱ_i on the blockchain can use the CID viewing function to access all verified CIDs for the current task. They then retrieve the corresponding $T_{iresult}$ data from the IPFS system and perform an off-chain computation using the pre-defined data integration algorithm. This integration calculation produces a list of binary arrays $< Ad_W, R >$, representing each worker's blockchain address and the corresponding incentive allocation. Once the integration calculation is complete, the worker uploads the result to the IPFS system, generating a new CID. Worker ϱ_i then applies a Keccak-256 hash function to this CID, obtaining a hash value H_d, which is uploaded to the computation result submission function. The smart contract then aggregates all submitted hash values into a set $S = \{H_{d1}, H_{d2}, \dots, H_{dm}\}$. Since all workers are incentivized to protect their interests and due to the irreversible nature of hash functions (making it difficult to deduce the original data from the hash), each worker is motivated to perform the integration computation to ensure their rewards. The details of the DWBF (Dynamic Weight Bounding box fusion) computation process will be elaborated in Section 4. The final data integration result is determined by the hash value H_{di} that appears most frequently in the submissions, according to the following formula:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 November 2024 doi:10.20944/preprints202411.2183.v1

$$\mathrm{H_{di}} = arg \max_{h \in S} count(h) \quad (3)$$

6. Incentive Distribution Stage: When the time reaches t_{end2} or the number of workers who submitted H_d reaches K, the smart contract transitions into the incentive distribution stage. At this stage, a designated on-chain worker ϱ_i must upload the original data, containing multiple binary arrays $< Ad_W, R >$ before the CID hash processing. This data represents each worker's address Ad_W and the corresponding incentive allocation for the task. The incentive distribution function within the smart contract will automatically execute the distribution of rewards to each worker W based on the pre-defined incentive allocation scheme.

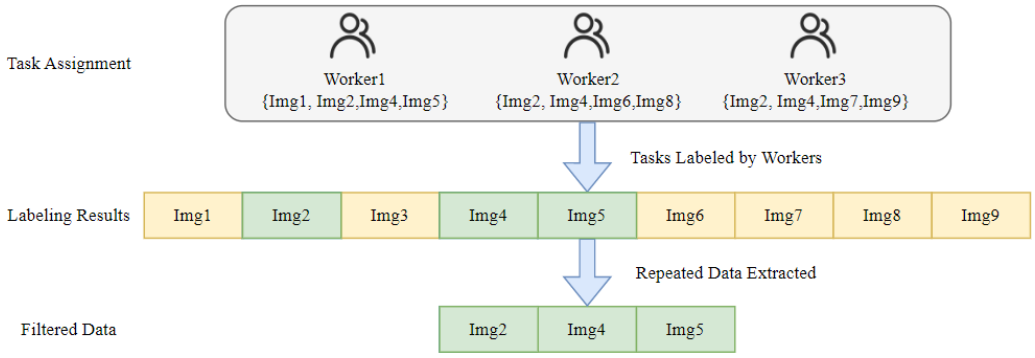

Figure 4. Illustration of the Annotation Quality Verification Method.

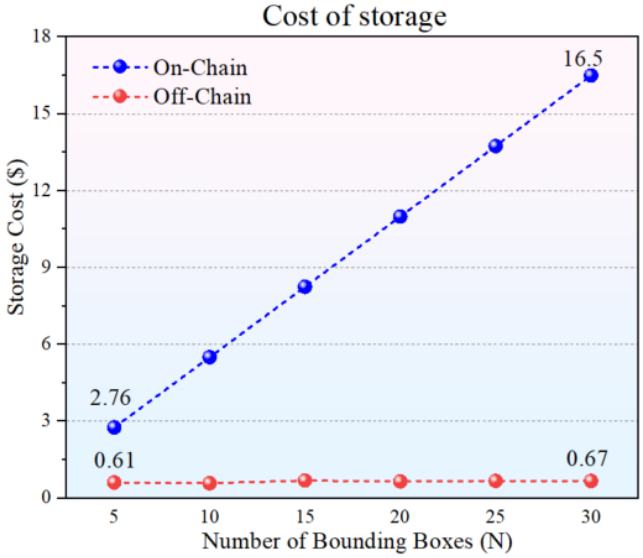

Figure 5. Comparison of On-Chain and Off-Chain Storage Costs.

4. Quality-Driven Incentive Mechanism

This section provides a detailed description of how the DWBF (Dynamic Intersection over Union Weighted Bounding Box Fusion) algorithm is used to verify the quality of bounding box annotations provided by workers. By evaluating each worker's contributions based on both workload and annotation quality, this approach enables the system to fairly distribute incentives, rewarding workers proportionally according to their performance. Section 4.1 introduces the bounding box annotation quality verification method, while Section 4.2 elaborates on the DWBF algorithm and the quality-driven incentive mechanism.

 doi:10.20944/preprints202411.2183.v1

4.1. Bounding Box Annotation Quality Verification Method

The requester incurs a cost for each annotated dataset; therefore, redundant annotations increase overall costs. To balance cost and benefit, this paper employs redundant annotation as a quality verification method. The core concept is to enhance the accuracy of the final annotations by increasing annotation redundancy. Specifically, the requester categorizes bounding box annotation tasks based on data complexity and accuracy requirements, identifying which data requires repeated annotations and how many repetitions, and assigns these tasks to various workers.

First, by introducing redundancy in annotations, the mechanism reduces individual annotation errors and biases. The DWBF algorithm assigns dynamic weights to each bounding box, producing more accurate results. Secondly, since workers are unaware of which data will undergo redundant annotation, they must treat each task seriously to avoid incentive loss, thus enhancing their accountability. Furthermore, by implementing a task stratification mechanism, which allocates worker resources based on data complexity and precision needs, annotation efficiency is improved.Finally, the quality-driven incentive mechanism, coupled with the DWBF algorithm, prevents the risk of unfairly undervaluing workers' contributions under subjective judgment. As shown in Figure 4 images such as Img2, Img4, and Img5 are annotated by two or more workers. For all data with repeated annotations, the collected results are processed through the DWBF algorithm described in Section 4.2 to determine the true bounding boxes and labels for each object in every image. This outcome is used to assess each worker's annotation quality and allocate incentives based on their performance.

4.2. DWBF Algorithm and Quality-Driven Incentive Mechanism

The CrowdBA architecture focuses on bounding box annotation, a method for identifying objects within images using rectangular boxes and labels. In such annotations, the location of each target object is defined by the coordinates of the rectangle's top-left corner (x_{min}, y_{min}) and bottom-right corner (x_{max}, y_{max}), while the category of the object is indicated by an associated label. The architecture employs a fixed encoding algorithm rather than advanced machine learning or deep learning algorithms. The primary reason for this choice is the consistency offered by fixed encoding algorithms, which generate identical outputs for the same input data. This consistency ensures that all participants produce the exact same hash values when operating on identical datasets. In contrast, machine learning or deep learning algorithms, due to random initialization and training processes, may yield slight variations in outputs even when applied to the same dataset. These minor discrepancies can lead to significantly different hash values, making it challenging to ensure result consistency. By storing the hash values of computed results on the smart contract and verifying them, the architecture guarantees transparency, fairness, and immutability in the evaluation process.

Typically, regions with the highest bounding box density are more likely to represent the true bounding box. To enhance accuracy and consistency in annotation tasks, this architecture introduces a Dynamic Intersection over Union Weighted Bounding Box Fusion (DWBF) algorithm. This algorithm integrates the consistency among annotators by assigning dynamic weights based on the Intersection over Union (IoU) metric. Specifically, the algorithm calculates the average IoU between each annotator's bounding box and those provided by all other annotators. Bounding boxes with higher IoU values are assigned greater weights, increasing their influence on the final result and reducing the impact of malicious annotations. This approach quantitatively measures annotation consistency among annotators. The calculation process is as follows:

1. IoU Calculation: Assume the coordinates of bounding box B_i are $(x_{i1}, y_{i1}, x_{i2}, y_{i2})$, and the coordinates of a bounding box C_j^l within a cluster C_j are $(x_{l1}, y_{l1}, x_{l2}, y_{l2})$, which correspond to $(x_{min}, y_{min}, x_{max}, y_{max})$. The Intersection over Union (IoU) value between each bounding box B_i and bounding boxes C_j^l within cluster C_j is calculated in three steps: First, calculate the coordinates of the intersection region using Equation (3). Next, calculate the area of the intersection region using Equation (4). Finally, calculate the IoU value using Equation (5):

 doi:10.20944/preprints202411.2183.v1

$$\begin{cases} x_{inter1} = max(x_{i1}, x_{l1}) \\ x_{inter2} = max(x_{i2}, x_{l2}) \end{cases} \quad \begin{cases} y_{inter1} = max(y_{i1}, y_{l1}) \\ y_{inter2} = max(y_{i2}, y_{l2}) \end{cases} \tag{3}$$

$$A_{inter} = max(0, x_{inter2} - x_{inter1}) * max(0, y_{inter2} - y_{inter1}) \tag{4}$$

$$IoU(B_i, C_j^l) = \frac{A_{inter}}{A_{union}} = \frac{A_{inter}}{A_{B_i} + A_{C_i} - A_{inter}} \tag{5}$$

2. Cluster Computation: Assume we have a set of annotated bounding boxes $\beta = \{B_1, B_2, \dots, B_n\}$. Each bounding box B_i needs to be assigned to a cluster. For each bounding box $B_i \in \beta$, we calculate its IoU value with all existing bounding boxes in each cluster. Suppose there are currently m clusters $\{C_1, C_2, \dots, C_m\}$, and each cluster C_j contains a set of bounding boxes $\{C_j^1, C_j^2, \dots, C_j^k\}$. For each bounding box $B_i \in \beta$, iterate through clusters C_j where the number of bounding boxes is at least κ, and calculate the IoU value with all bounding boxes in the cluster using Equation (6). Identify the maximum IoU value IoU_{ij}. If IoU_{ij} exceeds a specified threshold, add B_i to cluster C_j, otherwise, create a new cluster C_{m+1}.

$$IoU_{ij} = \max_l \left(IoU(B_i, C_j^l) \right) \tag{6}$$

3. IoU Dynamic Weight Calculation: For each bounding box B_i provided by annotator ϱ_i, the sum of IoU values with all other bounding boxes within cluster C_j is calculated pairwise. The final weight ω_i of ϱ_i is then computed using Equation (7). The bounding box with the highest weight in the cluster, C_j^i, is selected as its first member.

$$\omega_i = \frac{e^{e \sum_{j=1, j \neq i}^{N} IoU(B_i, B_j)}}{\sum_{k=1}^{N} e^{e \sum_{j=1, j \neq k}^{N} IoU(B_k, B_j)}} \tag{7}$$

where e is an exponential enhancement coefficient, N is the number of bounding boxes in cluster C_j.

4. Weighted Bounding Box Calculation: Let cluster C_j contain n bounding boxes, where the dynamic IoU weight score of the i-th bounding box is ω_i, and its coordinates are given by $[x_{min}^i, x_{max}^i, y_{min}^i, y_{max}^i]$. The coordinates of the weighted bounding box are calculated as follows in Equation (8):

$$\begin{aligned} x_{min} &= \frac{\sum_{i=1}^{N} x_{min}^i \omega_i}{\sum_{i=1}^{N} \omega_i} \\ x_{max} &= \frac{\sum_{i=1}^{N} x_{max}^i \omega_i}{\sum_{i=1}^{N} \omega_i} \\ y_{min} &= \frac{\sum_{i=1}^{N} y_{min}^i \omega_i}{\sum_{i=1}^{N} \omega_i} \\ y_{max} &= \frac{\sum_{i=1}^{N} y_{max}^i \omega_i}{\sum_{i=1}^{N} \omega_i} \end{aligned} \tag{8}$$

5. Label Calculation: For each cluster, a weighted summation is performed on the labels L_i of all bounding boxes within the cluster, along with their corresponding weights ω_i. The total weight score for each label is accumulated, and the label L_t with the highest cumulative weight is selected as the final label for the cluster. This process is described in Equation (9):

 doi:10.20944/preprints202411.2183.v1

$$L_t = \arg\max_{L} \sum_{\{i|L_i=L\}} \omega_i \tag{9}$$

6. Incentive Calculation: The incentive allocation for each worker aligns with their contribution and tiered incentive mechanism based on a composite score. First, the IoU and F1-score metrics are weighted in the evaluation of bounding boxes to compute each worker's composite score S_i using Equation (10). Workers are then grouped into different levels based on their composite scores, with incentives or penalties applied according to score ranges, as shown in Equation (11). If a worker's composite score $S_i < \psi_1$, they are classified as low-scoring and receive no incentive; If $\psi_1 \leq S_i < \psi_2$, they receive a partial incentive based on their score but face a progressive penalty; the lower the score, the higher the deduction, thus creating a positive incentive structure. If $S_i \geq \psi_2$, high-scoring workers receive both their earned incentive and a bonus from the penalty pool, encouraging consistent high-quality annotations. This structure encourages workers to maintain high annotation quality while ensuring fair and tiered incentive distribution based on performance metrics.

$$S_i = \alpha * IoU_i + \beta * \mathrm{F1}_{\mathrm{score}_i} = \alpha * IoU_i + \beta * \frac{2\mathrm{Precision}_i * \mathrm{Recall}_i}{\mathrm{Recall}_i + \mathrm{Recall}_i} \tag{10}$$

where α and β are weights for each metric, with $\alpha + \beta = 1$.

$$R_i = \begin{cases} R_i, & S_i < \Psi_1 \\ S_i * \left(1 - P * \left(\frac{\Psi_2 - S_i}{\Psi_2 - \Psi_1}\right)\right), & \Psi_1 \leq S_i < \Psi_2 \\ \left(\frac{S_i}{\sum_{j \in H} S_i}\right) * \left\{S_{total} + \left[\sum_{j \in M} S_k * P * \left(\frac{\Psi_2 - S_k}{\Psi_2 - \Psi_1}\right)\right]\right\}, & S_i \geq \Psi_2 \end{cases} \tag{11}$$

Where P is the maximum penalty rate, S_{total} is the total project incentive, H is the set of workers with scores $S_i \geq \Psi_2$, M is the set of workers with scores $\Psi_1 \leq S_i < \Psi_2$, Ψ_1 and Ψ_2 are the quality threshold values required by the project.

5. Experiments and Evaluation

5.1. Experimental Setup

For the system configuration, this study used the following hardware and software environment: an Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz processor, 16GB of RAM, and 1TB of storage, running on Windows 11 64-bit.

We developed smart contracts using Solidity 0.8.0 and compiled them in the Hardhat environment. To comprehensively evaluate and test the contract functionalities, we deployed them on Ethereum's Sepolia test network. Sepolia effectively simulates the Ethereum mainnet, ensuring the reliability and scalability of experimental results. For user interaction with the smart contracts, we used MetaMask, an Ethereum browser extension, allowing users to manage accounts and conduct transactions seamlessly via a browser. Connection to the Ethereum nodes was facilitated by Infura's service, which simplified access without the need to operate a full node, ensuring a stable network connection.During the experiments, a local IPFS daemon connected to the IPFS network was run. IPFS was used to store tasks and data, generating unique file hashes for data permanence and traceability. These files were persistently stored on the IPFS network through pinning.

This experimental setup enabled the evaluation of gas costs for the smart contracts, a comparison between off-chain IPFS storage and on-chain storage and computation in terms of resource usage, and a performance assessment of the DWBF algorithm against other algorithms, demonstrating the superiority of the proposed architecture.

 doi:10.20944/preprints202411.2183.v1

5.2. Smart Contract Execution Costs

Under the experimental setup outlined above, we deployed the smart contracts and measured their gas consumption to evaluate the CrowdBA framework's performance. Gas consumption was used as a metric to assess computational resource usage, testing and analyzing the contract's usability and efficiency.

In blockchain-based smart contract applications, gas consumption directly impacts efficiency and cost. To optimize the deployment process for task management contracts and avoid high costs associated with repetitive contract deployments by each requester, we adopted a factory contract model. By centralizing the deployment of the factory contract and task management contract, a single deployment suffices, significantly reducing overall gas consumption. Traditional task management contract deployment, typically involving complex logic, required up to 1,794,289 gas. However, the factory contract deployment cost was reduced to 610,596 gas. Subsequently, requesters only need to call the task group creation function (createTaskGroup), which further reduces gas consumption to 298,186 gas, eliminating the high costs of redeploying the entire task management contract.

Using the October 3, 2024, price of Ethereum (ETH) at $2,447.12 and a base gas fee of 4.62 gwei, we calculated the operational costs. The primary functions and their associated gas costs are shown in Table 3:

Table 3. Gas Consumption and Execution Costs for Contract Functions.

Contract	Function	Gas Used	ETH Value	Cost($)
Factory.sol	createTaskGroup	298186	0.00137761	3.37
TaskContract.sol	deposit	2848	0.00001315	0.03
	addWorker	52151	0.00024093	0.59
	submitHash	51381	0.00023738	0.58
	submitCID	13113	0.00006058	0.15
	submitResultHash	120999	0.00055901	1.37
	rewardDistribution	12000/person	0.00005544	0.14

The data in Table 2 indicates that the cost of calling each function within the smart contracts is well-controlled, with gas consumption remaining within reasonable limits. Given current ETH values, the contract's execution costs are relatively low, offering high economic efficiency and scalability in large-scale applications. The low gas consumption for individual functions is largely due to the shift of some storage and computation processes off-chain using the IPFS system for storage management. This architecture significantly reduces on-chain storage and computation burdens, effectively controlling the gas costs of function calls and enhancing overall performance and operational efficiency of CrowdBA.

5.3. On-Chain vs. Off-Chain Resource Consumption

In the execution of smart contracts, gas consumption serves as a key indicator of computational efficiency and economic cost. This study experimentally analyzed the gas consumption associated with processing varying amounts of data in both on-chain and off-chain operational modes.

As illustrated in Figure 5, there is a marked difference between on-chain and off-chain storage costs. On-chain storage costs increase linearly with the scale of data (number of bounding boxes), ultimately reaching $16.5 when the number of bounding boxes hits 30. In contrast, off-chain storage costs remain relatively stable, fluctuating only slightly between $0.61 and $0.67. This significant disparity is primarily due to the difference in storage methods. In a conventional on-chain storage model, the system must store all image IDs and their corresponding hashes, resulting in rapidly increasing resource consumption as data volume grows. However, with the off-chain storage solution used in this architecture, only the IPFS root hash (CID, Content Identifier) is stored on-chain. The CID is relatively short and fixed in length, and therefore gas consumption remains low, regardless of data

 doi:10.20944/preprints202411.2183.v1

volume. This approach significantly alleviates on-chain storage demands and reduces storage-related gas costs, enhancing cost-effectiveness, particularly when handling large-scale data.

Figure 6 compares on-chain and off-chain computation costs. As shown, the on-chain computation cost increases exponentially with the number of bounding boxes, reaching $174.71 when the count reaches 35. Conversely, the off-chain computa-tion cost remains stable at $1.88, unaffected by the number of bounding boxes. This disparity arises because increasing the number of bounding boxes leads to higher computational complexity and, subsequently, greater gas consumption for on-chain processing. In contrast, the off-chain approach uses a more efficient method. In this architecture, only two primary functions are needed: submitHash and submitResultHash, which upload the Keccak-256 hash of the computation results and the computation data itself. The submitHash function consumes 48,712 gas, while submitResultHash consumes 118,330 gas, totaling 167,042 gas. Since the hash length is fixed at bytes32, off-chain computation consumption remains con-stant regardless of bounding box quantity. Consequently, the off-chain computa-tion method effectively conserves on-chain computational resources, enhanced the scalability and cost-efficiency of CrowdBA.

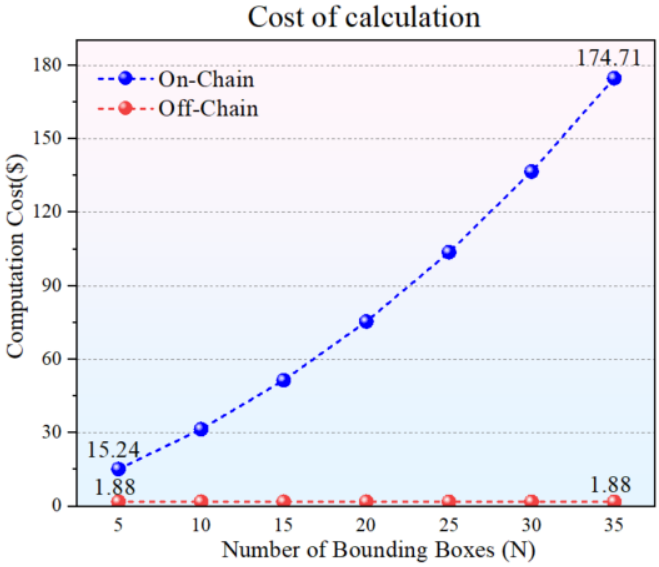

Figure 6. Comparison of On-Chain and Off-Chain Computation Costs.

The experimental results demonstrate that this combined on-chain/off-chain architecture substantially reduces gas costs. Compared to traditional on-chain processing, this architecture significantly lowers the storage and computation burden on the blockchain, avoiding cost increases associated with data scaling. Off-chain processing maintains a consistently low cost, while on-chain processing is minimized to essential confirmation operations. Thus, the overall architecture not only ensures security but also greatly enhances the economic efficiency and scalability of CrowdBA.

5.4. Comparison of Bounding Box Fusion Algorithms

This dataset is based on the Microsoft COCO (Common Objects in Context) dataset, a widely used dataset for computer vision tasks, primarily applied in object detection, segmentation, keypoint detection, and image captioning. COCO's main feature is its complex scenes with contextual information, where objects appear in diverse poses and perspectives and interact with other objects in real-world settings. This makes COCO one of the core benchmarks in computer vision research. For this experiment, we selected 2,369 images from the COCO dataset. Eleven deep learning object detection models were used to annotate these images with bounding boxes, selecting standard boxes with confidence scores above 0.5, and simulated additional bounding boxes to create a test dataset, while the ground truth bounding boxes for each image served as the validation set. The dataset covers 79 object categories, focusing on bounding box annotation. Each image contains between 10 and 125

 doi:10.20944/preprints202411.2183.v1

object bounding boxes, totaling 67,924 annotated boxes. This dataset allows a comprehensive evaluation of detection accuracy, bounding box overlap, and model robustness in varied scenarios.

Due to the crowdsourced nature of the bounding box annotations in CrowdBA, there is no concept of a "score" for each bounding box, meaning confidence scores are not available. Traditional Non-Maximum Suppression (NMS) algorithms typically select the bounding box with the highest score as the reference box and remove other boxes with high overlap. However, in the absence of score information in CrowdBA, it is not possible to prioritize boxes based on scores, necessitating a random strategy to select reference boxes. This study compares several post-processing algorithms for object detection, including Random Non-Maximum Suppression (Random-NMS), Random Non-Maximum Weighted (Random-NMW), and the DWBF algorithm at different IoU thresholds. Key evaluation metrics include Precision (reflecting false positive rates), Recall (reflecting missed detections), average F1-score, and mean IoU, assessing these algorithms' performance on the COCO dataset. The comparison of bounding box fusion algorithms is shown in Figure 7.

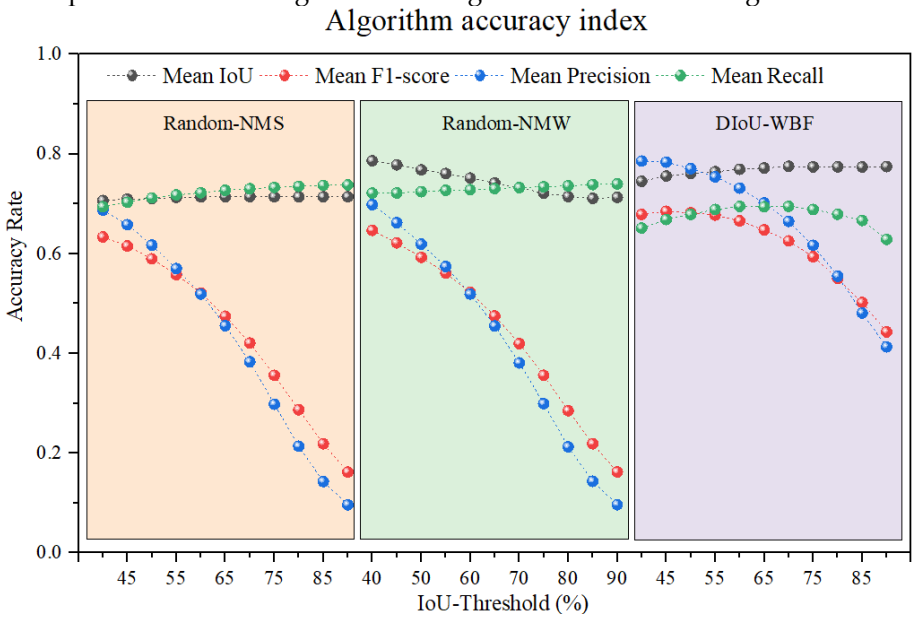

Figure 7. Comparison of Bounding Box Fusion Algorithms.

The results show that the DWBF algorithm demonstrates significant advantages across several key metrics under all threshold conditions:

- For Mean IoU, the DWBF algorithm consistently performs stably, particularly at higher IoU thresholds (70%-85%);
- Regarding Mean Precision, DWBF experiences minimal declines across different thresholds, performing notably well at low to mid-range thresholds (45%-65%);
- In terms of Mean Recall, DWBF is slightly lower than Random-NMS and Random-NMW but remains within an acceptable range;
- Overall, DWBF achieves a significantly higher Mean F1-score than the other methods, particularly at low to mid IoU thresholds.

These experimental results highlight DWBF's exceptional performance in object detection tasks, striking a balance between precision and recall by reducing missed detections while maintaining high accuracy. Additionally, the algorithm exhibits strong robustness in high IoU thresholds, providing more accurate estimations of bounding box overlap, making it well-suited for high-precision object detection in complex scenes.

6. Conclusions

The CrowdBA addresses the challenge of high costs in crowdsourced bounding box annotation on blockchain and introduces a fair incentive scheme tailored to bounding box annotation tasks. By integrating IPFS for off-chain storage and computation, the CrowdBA effectively alleviates issues of

 doi:10.20944/preprints202411.2183.v1

blockchain storage overload and computational constraints, ensuring data permanence and reduced costs. The DWBF algorithm, which dynamically assigns weights to bounding boxes based on IoU, infers true bounding boxes and accurately evaluates worker contributions. Through the implementation of related smart contracts, CrowdBA achieves fair and transparent incentive distribution, creating a low-cost, quality-driven environment for bounding box annotation tasks on the blockchain. Experimental analysis demonstrates the economic efficiency of this framework, significantly reducing gas consumption and showing that the DWBF algorithm outperforms traditional methods on key metrics such as Mean IoU, Precision, and F1-score. Future research may explore blockchain consensus algorithms [26–28] or Layer 2 scaling solutions [29–31] to further reduce on-chain operational costs and enhance the scalability of the CrowdBA architecture. Additionally, extending the annotation model to include speech labeling [32], semantic segmentation [33], and sentiment labeling [34] could provide more efficient and reliable data support for AI and machine learning applications.

Author Contributions: S.L. contributed to writing—original draft preparation, methodology, software, and validation of the proposed scheme; J.Z. contributed to writing—review and editing; R.G. contributed to supervision, project management, and funding acquisition. All authors have read and agreed to the final version of the manuscript.

Funding: This work is supported by the Fujian Provincial Science and Technology Project Guiding Project (2023H002) and the Quanzhou High-level Talent Innovation and Entrepreneurship Project (2022C00R).

Data Availability Statement: The data presented in this study are available on request from thecorresponding author. The data are not publicly available due to restrictions eg privacy.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu L, Ouyang W, Wang X, et al. Deep learning for generic object detection: A survey[J]. International journal of computer vision, 2020, 128: 261-318. [CrossRef]
2. Kaur J, Singh W. Tools, techniques, datasets and application areas for object detection in an image: a review[J]. Multimedia Tools and Applications, 2022, 81(27): 38297-38351. [CrossRef]
3. Whang S E, Roh Y, Song H, et al. Data collection and quality challenges in deep learning: A data-centric ai perspective[J]. The VLDB Journal, 2023, 32(4): 791-813. [CrossRef]
4. Aldoseri A, Al-Khalifa K N, Hamouda A M. Re-thinking data strategy and integration for artificial intelligence: concepts, opportunities, and challenges[J]. Applied Sciences, 2023, 13(12): 7082. [CrossRef]
5. Wylde V, Rawindaran N, Lawrence J, et al. Cybersecurity, data privacy and blockchain: A review[J]. SN computer science, 2022, 3(2): 127. [CrossRef]
6. Xia H, McKernan B. Privacy in Crowdsourcing: a Review of the Threats and Challenges[J]. Computer Supported Cooperative Work (CSCW), 2020, 29: 263-301. [CrossRef]
7. Jia L, Shao B, Yang C, et al. A Review of Research on Information Traceability Based on Blockchain Technology[J]. Electronics (2079-9292), 2024, 13(20). [CrossRef]
8. Alsadhan A, Alhogail A, Alsalamah H. Blockchain-Based Privacy Preservation for the Internet of Medical Things: A Literature Review[J]. Electronics, 2024, 13(19): 3832. [CrossRef]
9. Thakur K S, Ahuja R, Singh R. IoT-GChain: Internet of Things-Assisted Secure and Tractable Grain Supply Chain Framework Leveraging Blockchain[J]. Electronics, 2024, 13(18): 3740. [CrossRef]
10. Kochovski, P.; Gec, S.; Stankovski, V.; Bajec, M.; Drobintsev, P.D. Trust management in a blockchain based fog computing platform with trustless smart oracles. Future Gener. Comput. Syst. 2019, 101, 747–759. [CrossRef]
11. Sheehan K B. Crowdsourcing research: data collection with Amazon's Mechanical Turk[J]. Communication Monographs, 2018, 85(1): 140-156. [CrossRef]

 doi:10.20944/preprints202411.2183.v1

12. CrowdSFL: A secure crowd computing framework based on blockchain and federated learning[J]. Electronics, 2020, 9(5): 773. [CrossRef]
13. Yang S, Zhang Y, Cui L, et al. A Web 3.0-Based Trading Platform for Data Annotation Service With Optimal Pricing[J]. IEEE Transactions on Network Science and Engineering, 2023. [CrossRef]
14. Witowski J, Choi J, Jeon S, et al. MarkIt: a collaborative artificial intelligence annotation platform leveraging blockchain for medical imaging research[J]. Blockchain in Healthcare Today, 2021, 4. [CrossRef]
15. Zhu XR, Wu HH, Hu W. FactChain: A Blockchain-based Crowdsourcing Knowledge Fusion System. Journal of Software, 2022, 33(10): 3546-3564(in Chinese). http://www.jos.org.cn/1000-9825/6627.htm [CrossRef]
16. Li M, Weng J, Yang A, et al. CrowdBC: A blockchain-based decentralized framework for crowdsourcing[J]. IEEE transactions on parallel and distributed systems, 2018, 30(6): 1251-1266. [CrossRef]
17. Han X, Liu Y. Research on the consensus mechanisms of blockchain technology. Netinfo Security, 2017, 17(9): 147-152 (in Chinese with English abstract). [CrossRef]
18. Xu J, Wang C, Jia X. A survey of blockchain consensus protocols[J]. ACM Computing Surveys, 2023, 55(13s): 1-35. [CrossRef]
19. de Ocáriz Borde H S. An overview of trees in blockchain technology: merkle trees and merkle patricia tries[J]. University of Cambridge: Cambridge, UK, 2022. [CrossRef]
20. Taherdoost H. Smart contracts in blockchain technology: A critical review[J]. Information, 2023, 14(2): 117. [CrossRef]
21. Khan S N, Loukil F, Ghedira-Guegan C, et al. Blockchain smart contracts: Applications, challenges, and future trends[J]. Peer-to-peer Networking and Applications, 2021, 14: 2901-2925. [CrossRef]
22. Cai W, Wang Z, Ernst J B, et al. Decentralized applications: The blockchain-empowered software system[J]. IEEE access, 2018, 6: 53019-53033. [CrossRef]
23. Casino F, Dasaklis T K, Patsakis C. A systematic literature review of blockchain-based applications: Current status, classification and open issues[J]. Telematics and informatics, 2019, 36: 55-81. [CrossRef]
24. Liu Z, Luong N C, Wang W, et al. A survey on blockchain: A game theoretical perspective[J]. IEEE Access, 2019, 7: 47615-47643. [CrossRef]
25. Xu X, Weber I, Staples M, et al. A taxonomy of blockchain-based systems for architecture design[C]//2017 IEEE international conference on software architecture (ICSA). IEEE, 2017: 243-252. [CrossRef]
26. Wang W, Hoang D T, Hu P, et al. A survey on consensus mechanisms and mining strategy management in blockchain networks[J]. Ieee Access, 2019, 7: 22328-22370. [CrossRef]
27. Lashkari B, Musilek P. A comprehensive review of blockchain consensus mechanisms[J]. IEEE access, 2021, 9: 43620-43652. [CrossRef]
28. Ferdous M S, Chowdhury M J M, Hoque M A, et al. Blockchain consensus algorithms: A survey[J]. arXiv preprint arXiv:2001.07091, 2020. [CrossRef]
29. Gudgeon L, Moreno-Sanchez P, Roos S, et al. Sok: Layer-two blockchain protocols[C]//Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24. Springer International Publishing, 2020: 201-226. [CrossRef]
30. Gangwal A, Gangavalli H R, Thirupathi A. A survey of Layer-two blockchain protocols[J]. Journal of Network and Computer Applications, 2023, 209: 103539. [CrossRef]
31. Thibault L T, Sarry T, Hafid A S. Blockchain scaling using rollups: A comprehensive survey[J]. IEEE Access, 2022, 10: 93039-93054. [CrossRef]

 doi:10.20944/preprints202411.2183.v1

32. Malik M, Malik M K, Mehmood K, et al. Automatic speech recognition: a survey[J]. Multimedia Tools and Applications, 2021, 80: 9411-9457. [CrossRef]
33. Yu H, Yang Z, Tan L, et al. Methods and datasets on semantic segmentation: A review[J]. Neurocomputing, 2018, 304: 82-103. [CrossRef]
34. Bostan L A M, Klinger R. An analysis of annotated corpora for emotion classification in text[J]. 2018. [CrossRef]