electronics

MDPI

Article

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

Rongxin Guo *, Shenglong Liao and Jianqing Zhu

School of Engineering, Huaqiao University, Quanzhou 362000, China; 22014084009@stu.hqu.edu.cn (S.L.); jqzhu@hqu.edu.cn (J.Z.)
* Correspondence: grxeee@hqu.edu.cn

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 the 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

check for updates

Academic Editor: Fabio Grandi

Received: 20 November 2024
Revised: 31 December 2024
Accepted: 11 January 2025
Published: 17 January 2025

Citation: Guo, R.; Liao, S.; Zhu, J. CrowdBA: A Low-Cost Quality-Driven Crowdsourcing Architecture for Bounding Box Annotation Based on Blockchain. *Electronics* **2025**, *14*, 345. https://doi.org/10.3390/electronics14020345

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

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 enhance the model's ability to learn and interpret visual information efficiently [1,2]. In the bounding box annotation task, the position of each object within an image is defined by four coordinates (x_{min}, x_{max}, y_{min}, y_{max}), and each object is assigned a corresponding class label (label). For instance, in an image containing multiple cars, the bounding box coordinates and the class label (e.g., "sedan", "SUV") for each vehicle are accurately annotated and stored. This standardized annotation approach ensures data consistency and structure, facilitating subsequent model training and validation processes. 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 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, compared to existing blockchain-based crowdsourcing platforms, the research presented in this 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 a 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].

Compared to existing blockchain-based crowdsourcing platforms, this study presents several key differences. First, many current platforms adopt consortium blockchain architectures to reduce storage and computation costs. However, the centralized nature of consortium blockchains limits the decentralization and transparency of the platform. In contrast, the proposed CrowdBA architecture is built on the Ethereum public blockchain, fully leveraging the decentralization and transparency advantages of blockchain technol-

ogy. In traditional frameworks, IPFS is commonly used as a tool for file storage between trusted parties, where the requester only needs to confirm that the file has been uploaded to the IPFS system without verifying the quality of the uploaded content; this is also one of the pain points of the IPFS system, the inability to exchange information with the blockchain. However, in crowdsourcing scenarios, the quality of annotations is critical to the overall outcome of the task. Therefore, we argue that it is essential to design a robust architecture and process to ensure that the annotations uploaded by each worker to the IPFS system are valuable and that their quality can be effectively assessed. Moreover, existing quality-driven incentive mechanisms typically rely on manual reviews to verify annotation quality, which not only increases operational costs but also introduces potential subjective bias, impacting the fairness of the incentives. To address this issue, this study proposes the Dynamic Intersection over union-weighted bounding box fusion (DWBF) algorithm. The algorithm automatically evaluates annotation quality, avoiding the high costs and potential bias associated with manual reviews while making the incentive distribution more transparent and fair. The DWBF algorithm dynamically calculates the annotation quality of each worker, using this as the basis for incentive distribution, thereby reducing the interference of human factors and effectively preventing the requester from unfairly lowering the worker's labor value.

This paper implements 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 store the actual annotation data on IPFS while only storing the CID (Content Identifier) of the data on the blockchain. This approach offloads storage and computation off-chain, thereby mitigating the high costs associated with data storage and computation on the blockchain and enhancing the scalability of the system.
- 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 utilize a dataset consisting of bounding boxes annotated by 11 deep learning models and some computer-simulated annotations, with the gold standard bounding boxes from the COCO dataset serving as the validation set. The accuracy of the proposed DWBF algorithm is evaluated, 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 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 are 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].

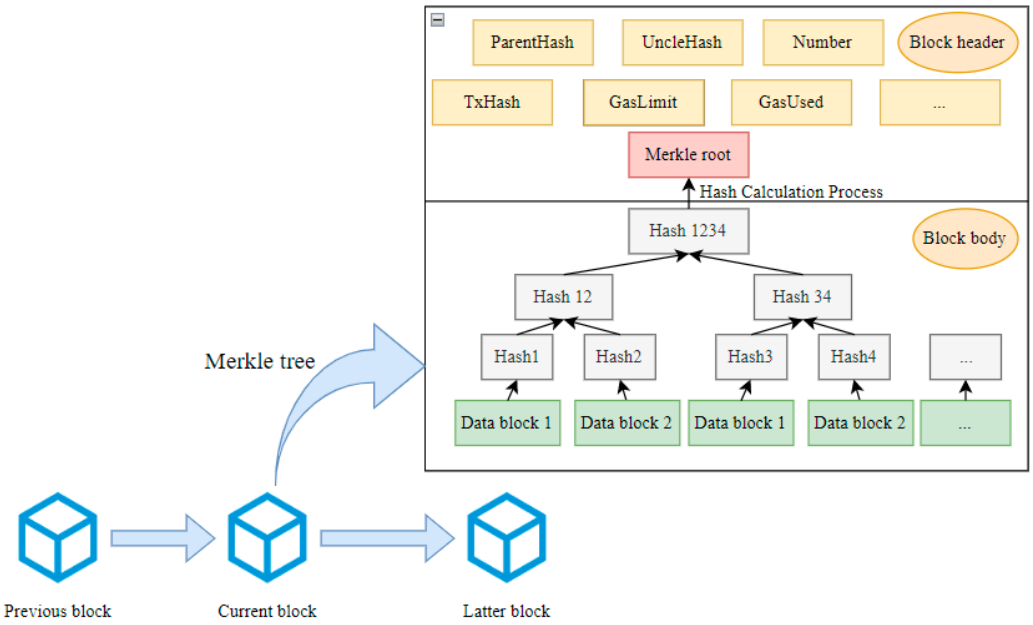


Figure 1. Block structure.

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 least transparency, as data remain internal, posing a risk of internal abuse. Given that 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	1000–10,000 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

This paper aims to develop a low-cost, quality-driven crowdsourcing architecture for bounding box annotation based on the Ethereum public blockchain, addressing the issue of insufficient labeled data in large-scale machine learning and artificial intelligence model training. 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 and become 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. Blockchain and smart contracts: The blockchain acts as an on-chain storage server in the system. The code running on the Ethereum-based smart contract ensures the transparency of algorithmic rules and the automatic distribution of incentives. Smart contracts are responsible for recording and validating data, automatically executing predefined incentive distribution rules, and managing the processes of task publication and acceptance.
4. IPFS: IPFS (InterPlanetary File System) serves as an off-chain storage solution within the system. It is a distributed protocol designed for the storage, sharing, and retrieval of files. 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.
5. Computation Script: The system provides a dedicated script responsible for computation, with the script's code made publicly available for auditing. The algorithm within the script is the DWBF bounding box fusion algorithm designed in this study. This script offers computational support to both the requester and workers. By utilizing off-chain computation in this manner, the system reduces the computational overhead on the Ethereum network, thereby saving on ETH resources.

6. The system workflow is as follows: The requester first creates a smart contract that includes the task terms and incentive mechanism, then deploys it to the Ethereum network and funds it with incentives. Upon applying for a task, data annotation workers must complete a test task and submit it to the requester. After review, the requester grants the worker annotation qualifications. Once the worker completes the data annotation, the results are uploaded to the IPFS system, generating a unique Content Identifier (CID). The hash value of this CID is then returned to the smart contract for recording. Upon task completion, the worker publishes the CID. Both the requester and the worker can view the results and execute the DWBF bounding box fusion algorithm by calling an off-chain computation script. This process evaluates the annotation results, calculates each worker's contribution and incentive values, and records the hash value of the evaluation results on the smart contract. The computed results are then made public in the next phase. Based on the worker's contribution value, the smart contract automatically executes incentive distribution, ensuring a fair and equitable reward allocation. Meanwhile, the requester can also view the task results through the IPFS system.

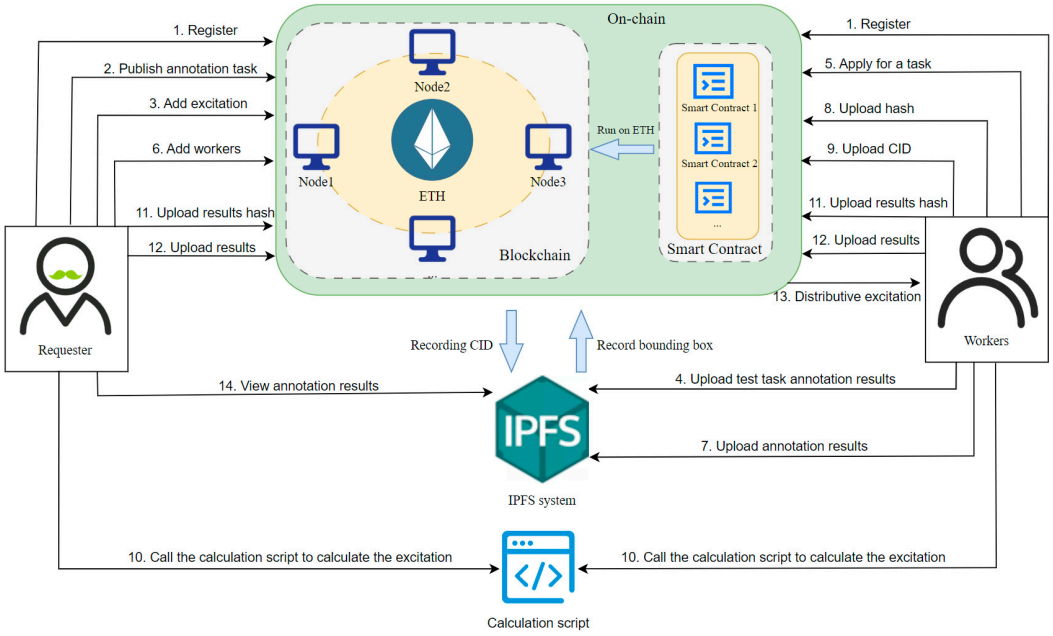


Figure 2. CrowdBA architecture.

In this paper, it is assumed that the requester D_i has a set of annotation tasks $T = \{\tau_1, \tau_2, \ldots, \tau_m\}$, where each task τ corresponds to a single image, where τ_i represents the i-th annotation task, corresponding to the i-th image that needs to be annotated by a worker. The requester deploys a contract instance C through a contract factory. 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, G	Data submission deadline, reward calculation deadline, maximum number of workers, worker's deposit
10	Desc_T	Task description
11	Ad_T, Ad_R, Ad_W	Task address, requester address, worker address
12	H_d	Data hash

3.2. Security Analysis

The CrowdBA architecture faces several security threats, including resilience to network failures and attacks, protection against external data theft, defense against freeloading attacks, and mitigation of Sybil attacks. This section provides a detailed analysis of these threats to demonstrate the security of the architecture.

1. Network fault and attack resistance: The CrowdBA system is built on the Ethereum public blockchain, which inherently possesses decentralization features. This decentralization endows the system with advantages such as fault tolerance and robustness, which are characteristic of blockchain technology. Even if some nodes in the system fail, the overall crowdsourcing service remains largely unaffected, thereby effectively preventing network failures and mitigating typical malicious network attacks.
2. Protection against external data theft: In the context of crowdsourced bounding box annotation, external attackers would need simultaneous access to both the original images and the corresponding annotation data to obtain sensitive information. Without access to the original images, merely accessing the bounding box locations and labels does not provide attackers with meaningful or complete data. In this architecture, image data are uploaded by requesters to the IPFS system. The returned Content Identifier (CID) is encrypted using the worker's public key before being sent to them. Workers then decrypt the CID using their private key, ensuring that the image data CID does not appear in plaintext on the blockchain.
3. Worker incentive protection: In this architecture, the incentive distribution process is automated through smart contracts, with the rules for allocation determined by the predefined DWBF algorithm. All workers upload the computed results, and the incentives are distributed accordingly. The requesters can only offer incentives to attract workers during the contract creation phase and do not have the authority to alter the distribution of rewards. This ensures that workers' interests are protected and that the distribution of incentives is fair and equitable.
4. Resistance to free-riding attacks: To prevent workers from exploiting others' annotations and bypassing the qualification process, the system ensures that workers encrypt task data using the requester's public key. Since the requester's private key is not available, other workers cannot access the encrypted data, thus preventing free-riding attacks. During the task submission phase, workers upload only the encrypted hash of

their annotations, not the raw data. Additionally, workers are required to provide the hash of their annotations upon submission, ensuring that each worker's annotation remains traceable. In the verification phase, workers must publicly disclose their annotation results; however, since no new data can be submitted at this stage, the integrity of previously uploaded annotations is protected, effectively mitigating the risk of malicious modification and free-riding attacks.

5. Sybil attacks: Our approach reduces the risk of Sybil attacks by leveraging redundant annotations, which improves the quality of the results. In our system, each task is annotated by multiple distinct workers, with the tasks assigned to each worker being partially overlapping rather than identical. As a result, even if an attacker creates multiple fake identities to participate in the task, the annotations they provide are likely to differ due to the task distribution. This leads to a lower overall score for the attacker. In other words, if an attacker attempts to use Sybil identities to submit multiple incorrect annotations, these annotations are unlikely to receive high scores, causing the attacker to lose their stake.

3.3. 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.

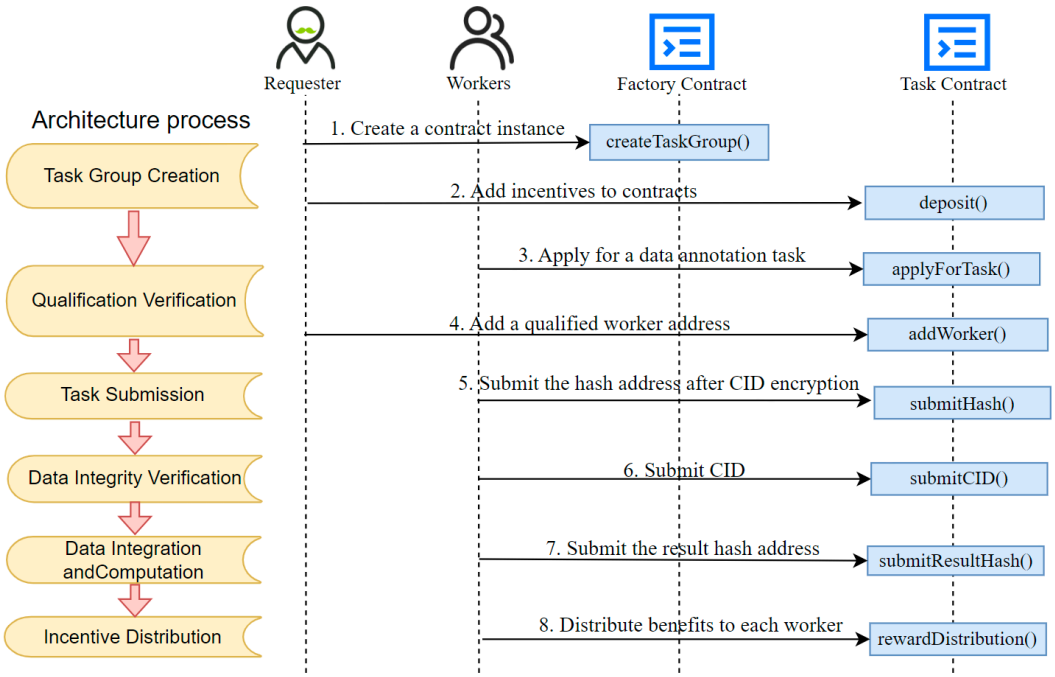


Figure 3. Smart contract workflow diagram.

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:

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

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 $< Img_{ID}, label, x_{min}, y_{min}, x_{max}, y_{max} >$, which is uploaded to the IPFS system. The returned CID is encrypted using the public key of D and then handed over to the task request function. D can use their private key to decrypt the CID and obtain the corresponding IPFS address of T_{iresult} for viewing. 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 Function 1, 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 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 (bool)
1.  if (msg.sender == owner) then
2.    if (currentEligibleWorkers < maxWorkers) then
3.      [eligibleWorkers] ← workerAddress
        currentEligibleWorkers ← currentEligibleWorkers + 1
        workerAdded ← true;
4.  else
5.  Return error;
6.  else
7.  Return error;
8.  end if
9.  return workerAdded
```

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 first performs an availability check on the IPFS network by querying the node's status to confirm its operational state. This includes verifying the node's connectivity, health status, and storage capacity to ensure that tasks can be successfully uploaded and accessed. Subsequently, a set of images and the corresponding one-dimensional annotation array $< Img_{ID} >$ for the task T_i are uploaded to the IPFS system. The returned CID is then encrypted using the requester's public key and passed to the annotator. The annotator can decrypt the CID using their private key to obtain the IPFS address corresponding to task T_i and proceed with the 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 are divided into 256 KB blocks, $Data = \{B_1, B_2, \ldots, B_n\}$. 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. Subsequently, the worker processes the CID using the Keccak-256 hashing algorithm to obtain a secure hash value, which is submitted to the smart contract only after the worker has deposited the required task collateral. If the worker fails to provide the necessary collateral, the function to submit the hash value will be inaccessible. Additionally, the contract will not allow submissions once the predefined time limit has expired or once the minimum number of required submissions has been reached. The uploaded hash values will serve as the primary basis for validating data integrity in the next stage, ensuring that the data remain untampered and authentic. This method strengthens the security of the system by preventing malicious alterations and ensuring the accuracy of the annotation results submitted by workers.

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

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 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 Function 2. 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. 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

```
1.  if (currentTime ≥ timeEnd1) then
2.    if (maxWorkers ≥ submittedWorkers) then
3.    if (WorkerSubmissions[msg.sender] = NULL) then
4.    if (Keccak-256(CID) = H_d) then
5.    [CIDSubmission] ← CID
6.    submittedWorkers + 1
7.              CIDSubmitted ← ture;
8.    else
9.    Return error;
10.   else
11.   Return error;
12.   else
13.   Return error;
14.   else
15.   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 off-chain computation using the predefined data integration algorithm in the computational script. 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 $S = \{H_{d1}, H_{d2}, \ldots, H_{dm}\}$. 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 and Function 3:

Function 3: getMostVotedDataHash

input: dataHash
output: TaskSubmitted ← true

if (currentTime ≥ rewardCalculateDeadline) **then**
Return error: "Current time is after deadline"
else
if (taskData (0).submissionCount ≥ maxRewardCalculateSubmissions) **then**
Return error: "Maximum submissions reached"
else
task ← taskData(0)
if (task.submitters[msg.sender] = NULL) **then**
task.submitters[msg.sender] ← dataHash
task.votes[dataHash] ← 1
task.submissionCount ← task.submissionCount + 1
if (task.votes[dataHash] > task.currentHighestVotes) **then**
task.currentMostVotedHash ← dataHash
task.currentHighestVotes ← task.votes[dataHash]
end if
TaskSubmitted ← true
Emit TaskSubmitted(msg.sender, 0, dataHash)
else
Return error: "You have already submitted a result."
end if
end if
end if

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. These data represent 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 predefined incentive allocation scheme.

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.

4.1. Bounding Box Annotation Quality Verification Method

The requester is required to pay for each annotation task, and increasing redundant annotations directly increases costs. Considering the trade-off between cost and benefit, this paper argues that task allocation should not be determined by the system but rather be decided by the requester. Based on the complexity of the data and the required accuracy, the requester divides the annotation task into multiple segments, specifying which data should be annotated redundantly and the number of redundant annotations required. Specifically, if the accuracy requirement for the annotation task is low, only a small portion of the task can be designated for redundant annotations. Conversely, if high accuracy is required, more redundant annotations can be specified to verify and improve the accuracy. During the qualification verification stage, annotators inform the requester of the number of tasks they are willing to undertake, with the content of the tasks being defined by the requester.

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. 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.

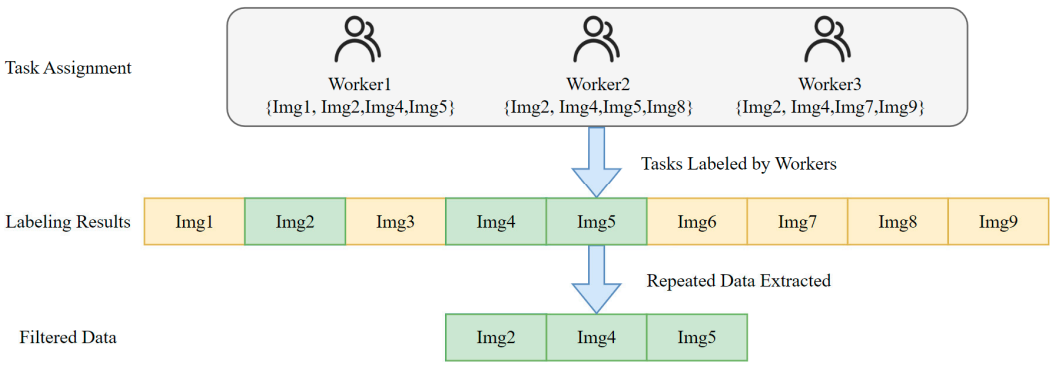


Figure 4. Illustration of the annotation quality verification method.

4.2. DWBF Algorithm and Quality-Driven Incentive Mechanism

The CrowdBA architecture focuses on the bounding box annotation task, which involves using rectangular boxes and labels to identify objects within images. This type of annotation requires determining the position and corresponding category of each target object, where the position is described by the coordinates of the top-left corner (x_{min}, y_{min}) and the bottom-right corner (x_{max}, y_{max}) of the rectangle, while the category information is provided through an attached label. Wang L, through modeling and analysis of pricing strategies on TopCoder, explored the impact of incentive mechanisms on annotation quality and concluded that an appropriate reward strategy can enhance task annotation quality, while excessively high rewards may lead to a decline in quality. Therefore, understanding the rationale behind incentive mechanisms is crucial for optimizing the quality of crowdsourced annotation tasks [26]. A significant challenge in crowdsourced data annotation systems is how to properly assess the quality of worker annotations and fairly distribute rewards based on annotation quality. To address this, we present a Dynamic Intersection over a union-weighted bounding box fusion (DWBF) algorithm. This approach can comprehensively analyze the semi-trusted bounding boxes obtained from the crowdsourcing system and infer a relatively reliable ground truth bounding box. This inferred ground truth serves as a basis for incentive distribution among workers.

The DWBF algorithm opts for a fixed encoding method rather than more advanced machine learning or deep learning techniques primarily due to its ability to produce consistent output results for identical input data. This consistency ensures that, when run on the same dataset, all participants will compute identical hash values. In contrast, machine learning or deep learning algorithms, due to their random initialization and training processes, may generate subtle differences in output even when operating on the same dataset. These minute variations can result in significantly different hash values, making it difficult to guarantee consistency in the results. By storing the hash values of the computational outcomes in a smart contract and verifying them, the process ensures transparency, fairness, and immutability in the evaluation.

Hard-coded bounding box fusion methods such as Non-Maximum Suppression (NMS) and Non-Maximum Weighting (NMW) rely on the bounding boxes and confidence scores output by deep learning models for fusion. The score parameter is crucial in these algorithms to assess the "importance" of a bounding box, ensuring that the selected or fused box most accurately represents the target. However, in crowdsourcing scenarios, the annotation process is performed manually, and no confidence score is available. Consequently, traditional methods typically adopt two strategies for bounding box fusion: one assumes all bounding boxes are of equal importance and selects the first bounding box using a random strategy. However, this approach overlooks the variations in skill levels and annotation quality among different workers, which can lead to fusion results that are influenced by low-quality annotations that cannot be identified, thereby reducing the overall accuracy of the annotations. Hard-coded bounding box fusion methods such as NMS (Non-Maximum Suppression) and NMW (Non-Maximum Weighting) rely on the bounding boxes and their confidence scores output by deep learning models to perform fusion. The score parameter is crucial in these algorithms, as it helps assess the "importance" of a box and ensures that the selected or fused boxes are most likely to accurately represent the target. However, in crowdsourcing scenarios, the annotation process is manually performed, and no confidence score is provided. As a result, traditional methods typically treat all bounding boxes as equally important, often employing a random strategy to select the first bounding box. This approach overlooks the variations in the skill levels and annotation quality among different workers, leading to fusion results that may be influenced by low-quality annotations, which are undetectable, thereby reducing the overall annotation accuracy.

In general, regions with the highest concentration of bounding boxes are more likely to represent the true bounding box. To enhance the accuracy and consistency of annotation tasks, the DWBF bounding box fusion algorithm takes into account the consistency among annotators by calculating the Intersection over Union (IoU) values between annotation boxes. The algorithm dynamically weights each annotation box based on these IoU values to perform the fusion of bounding boxes. The computational process of the DWBF bounding box fusion algorithm proposed in this framework is as follows:

7. 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):

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

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

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

8. Cluster computation: Assume we have a set of annotated bounding boxes $\beta = \{B_1, B_2, \ldots, B_n\}$. 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 $\{C_1, C_2, \ldots, C_m\}$, and each cluster C_j contains a set of bounding boxes $\left\{C_j^1, C_j^2, \ldots, C_j^k\right\}$. 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}.

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

9. 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.

10. 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 $[x^i_{min}, x^i_{max}, y^i_{min}, y^i_{max}]$. The coordinates of the weighted bounding box are calculated as follows in Equation (8):

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

11. 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):

$$L_t = \underset{L}{\operatorname{argmax}} \sum_{\{i|L_i=L\}} \omega_i \tag{9}$$

12. 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 R_i applied according to score ranges, as shown in Equation (11). If a worker's composite score is below the required threshold ψ_1, they are classified as low-scoring and receive no incentive, and the previously paid task deposit will not be refunded. If their score falls between ψ_1 and ψ_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. Additionally, the previously paid task deposit will be refunded. If their score exceeds $\geq \psi_2$, 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 $\alpha + \beta = 1$.

$$R_i = \begin{cases} 0, S_i < \Psi_1 \\ S_i * \left(1 - P * \left(\dfrac{\Psi_2 - S_i}{\Psi_2 - \Psi_1}\right) * \dfrac{\mathrm{N_i}}{\mathrm{N_{total}}}\right) + \mathrm{G_i}, \Psi_1 \leq S_i < \Psi_2 \\ \left(\dfrac{S_i}{\sum\limits_{j \in H} S_j}\right) * \left\{S_{total} + \left[\sum_{k \in M} S_k * P * \left(\dfrac{\Psi_2 - S_k}{\Psi_2 - \Psi_1}\right)\right] * \dfrac{\mathrm{N_i}}{\mathrm{N_{total}}}\right\} + \mathrm{G_i}, S_i \geq \Psi_2 \end{cases} \tag{11}$$

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. The total number of bounding boxes received for the task is denoted as $\mathrm{N_{tatol}}$, and $\mathrm{N_i}$ represents the

number of bounding boxes annotated by worker i, with G_i representing the task deposit staked by the user.

The DWBF algorithm can be broadly divided into two main steps: cluster calculation and dynamic weight-based bounding box fusion. The cluster calculation step determines the number of valid bounding box objects within an image. The dynamic weight-based bounding box fusion step calculates the ground truth for each bounding box in the image. The pseudocode for the cluster calculation process is provided in Algorithm 1, with the detailed procedure outlined as follows: First, two empty lists are initialized: clusters, which store information about all the clusters, and cluster_indices, which store the index of the cluster to which each bounding box belongs. The function then begins by iterating over each image in the dataset (denoted as img_id). For each image, the function filters out all the bounding boxes corresponding to that image from the data and stores their labels and coordinate information in a two-dimensional array, such as boxes. Next, the function initializes an empty list, clusters_for_img, for the current image to store all the assigned clusters. The function then iterates over all existing clusters in the image, calculating the Intersection over Union (IoU) between the current bounding box and all bounding boxes within each cluster using the formulas (3), (4), and (5). If the IoU of a bounding box within a cluster exceeds a predefined threshold (iou_threshold), the current bounding box has been matched to a cluster. If the current bounding box does not match any existing cluster, the function creates a new cluster and adds the bounding box to it. After processing all bounding boxes, the function adds all clusters of the current image to the clusters list and records the corresponding cluster index (i.e., the cluster number) for each bounding box in the cluster_indices list. Finally, the function returns these two lists: clusters and cluster_indices, which represent all the clusters to which the bounding boxes belong and the respective indices of those clusters. This process ensures that bounding boxes with high similarity (as measured by IoU) are grouped together in the same cluster, facilitating subsequent processing and analysis.

Algorithm 1: clusterCalculate

Input: img_id, box[label, x_{min}, x_{max}, y_{min}, y_{max}], iou_threshold
Output: img_id, box[label, x_{min}, x_{max}, y_{min}, y_{max}], cluster

clusters[] ← Null
clusters_indices[] ← Null
for each img_id in data **do**
boxes ← box[label, x_{min}, x_{max}, y_{min}, y_{max}]
clusters_for_img[] ← Null
for each box in Boxes **do**
for each boxes and other box in Boxes **do**
iou ← Equations (3)–(6)
if iou > iou_threshold **do**
cluster[] ← box[label, x_{min}, x_{max}, y_{min}, y_{max}]
break
else if iou < iou_threshold **do**
new_cluster[] ← box[label, x_{min}, x_{max}, y_{min}, y_{max}]
clusters_for_img[] ← new_cluster[]
end if
end for
end for
end for
for each cluster **do**
clusters[] ← cluster[]
cluster_indices[] ← box[[i] * len(cluster)]
end for
return img_id, box, cluster

The process for dynamically weighted bounding box fusion is outlined in the pseudocode in Algorithm 2. The detailed procedure is as follows: First, for each image (img_id) and its corresponding clusters, the algorithm iterates through each bounding box and computes its weight (ω) using Equation (7). This weight is typically calculated based on the Intersection over Union (IoU) between bounding boxes. Then, using Equation (8), a weighted average is computed for the coordinates (x_{min}, x_{max}, y_{min}, y_{max}) of all the bounding boxes within the cluster, resulting in the coordinates of the fused bounding box, where boxes with higher weights contribute more significantly to the final result. Next, Equation (9) is used to calculate the weighted label, with the highest-scoring label selected as the representative label for the cluster. Finally, the algorithm outputs the coordinates and labels for all bounding boxes in each image. This process ensures that bounding boxes with higher confidence, as reflected by their weights, have a greater influence on the final fused bounding box and label, thereby improving the overall accuracy of the annotation.

Algorithm 2: Box calculate

Input: img_id, box[label, x_{min}, x_{max}, y_{min}, y_{max}], cluster
Output: img_id, box[label, x_{min}, x_{max}, y_{min}, y_{max}]

for each img_id in data **do**
for each cluster in img_id **do**
for each box in cluster **do**
$\omega \leftarrow$ Equation (7)
box[label, x_{min}, x_{max}, y_{min}, y_{max}] $\leftarrow \omega$
end for
box[x_{min}, x_{max}, y_{min}, y_{max}] $\leftarrow$ Equation (8)
label $\leftarrow$ Equation (9)
box[x_{min}, x_{max}, y_{min}, y_{max}] $\leftarrow$ label
end for
end for

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, 16 GB of RAM, and 1 TB of storage, running on Windows 11 64-bit.

In this experiment, the bounding box fusion algorithm was implemented using Python, while Solidity 0.8.0 was chosen as the programming language for the smart contract, which was compiled within the Hardhat environment. To thoroughly evaluate and test the functionality of the smart contract, it was deployed on the Sepolia test network of Ethereum. Sepolia serves as an effective simulation of the Ethereum mainnet, ensuring the reliability and scalability of the experimental results. To facilitate interaction with the smart contract, the MetaMask Ethereum browser plugin was used, allowing users to easily manage accounts and perform transactions directly through the browser. The Ethereum node was connected via services provided by Infura, which eliminated the complexity of running a full node while ensuring seamless network connectivity. For storage, a local IPFS daemon was run to connect to the IPFS network. IPFS was used to store tasks and data, generating unique IPFS file hashes to ensure data persistence and traceability. These files were persistently stored through "pinning" on the IPFS network, guaranteeing their long-term availability.

13. Experimental dataset: The dataset used in this experiment is based on the Microsoft COCO (Common Objects in Context) dataset, which is widely employed in computer vision tasks such as object detection, segmentation, keypoint detection, and image captioning. Its primary feature is the provision of complex scenes containing contextual

information, where objects appear not only in diverse poses and viewpoints but also interact with other objects in real-world settings. This makes the COCO dataset one of the core benchmarks for research in the field of computer vision. For this experiment, 2369 images were selected from the dataset. Eleven deep-learning object detection models were used to perform bounding box annotations on these images. Bounding boxes with a confidence score above 0.5 were selected as standard boxes, and a subset of these boxes was simulated for testing. The ground truth bounding boxes for each image were used as the validation set. The dataset covers 79 object categories and focuses on the bounding box annotation task. The number of object bounding boxes per image ranges from 10 to 125, with a total of 67,924 bounding boxes annotated.

14. Experimental parameters: As of 20 December 2024, the price of Ether (ETH) was USD 3121.96, with a base gas fee of 8.1 gwei and a gas limit of 3,000,000. The Tips were set to 0. All transaction fee calculations in this experiment were based on these values.
15. Experimental evaluation metrics: The experiment evaluated the execution cost of the functions within the developed smart contract. It compared the resource consumption between off-chain IPFS storage and on-chain storage and computation, including testing the upload and download speeds of IPFS. Finally, the accuracy of the DWBF algorithm was compared with that of other algorithms.

5.2. Smart Contract Execution Costs

In the experimental setup described above, the performance of the proposed CrowdBA framework was evaluated by deploying smart contracts and testing their associated cost overheads. The use of computational resources was measured through gas consumption during function execution, with the contracts being tested and analyzed to assess their usability. In the context of blockchain smart contract applications, gas consumption directly impacts both efficiency and cost. To optimize the deployment process of the task management contract and avoid the repetitive deployment of high-cost contracts by each requester, this paper introduces the factory contract model. By centrally deploying the factory contract (617,602 gas) and the task management contract (2,536,131 gas), a single deployment operation is required, thus reducing overall gas consumption. With the factory contract model in place, subsequent requesters only need to invoke the task group creation function, createTaskGroup, to create their own data annotation tasks, thereby avoiding the high costs of redeploying the entire task management contract. This paper tests the gas consumption of the smart contract during both the first and subsequent function calls, as well as the overall system expenditure with varying numbers of workers.

The gas consumption for the first function call and execution costs are presented in Table 3. It can be observed that the invocation costs for the individual functions in the smart contract were effectively controlled, with gas consumption remaining within reasonable limits. While the cost of invoking the task creation function in the factory contract is high, it is borne only once by the requester when initiating a new task. Other functions within the smart contract exhibit relatively low execution costs. The reason for the low gas consumption of the various functions is that certain storage and computational operations are offloaded to off-chain management, specifically utilizing the IPFS system for storage. This architecture significantly reduces the storage and computational load on the blockchain, markedly lowering the consumption of on-chain resources and thus effectively controlling the gas costs associated with function invocations. This, in turn, improves the overall performance and operational efficiency of CrowdBA.

Table 3. Gas consumption and execution costs for contract functions.

Contract	Function	Gas Used	ETH Value	Cost ($)
Factory.sol	createTaskGroup	365,116	365,116	9.23
TaskContract.sol	deposit	10,337	27,437	0.26
	applyForTask	6566	6566	0.17
	addWorker	52,063	34,963	0.88
	submitHash	75,921	58,821	1.92
	submitCID	58,963	41,861	1.49
	submitResultHash	121,137	41,815	3.06
	rewardDistribution	14,000/person	14,000/person	0.35

The gas consumption for subsequent function calls and execution costs is shown in Table 4. It can be observed that the cost of invoking individual functions in the smart contract decreases after the initial call. This is primarily because the first execution typically involves one-time tasks such as initialization and state storage, which require higher gas consumption. In contrast, subsequent calls benefit from the ability to leverage the previously stored state and optimizations, leading to a reduction in computational and storage operations, thereby effectively lowering gas consumption.

Table 4. Gas consumption for subsequent function executions.

Contract	Function	Gas Used	ETH Value	Cost ($)
TaskContract.sol	deposit	27,437	0.0002222397	0.69
	applyForTask	6566	0.0000531846	0.17
	addWorker	34,963	0.0002832003	0.88
	submitHash	58,821	0.0004764501	1.49
	submitCID	41,861	0.0003390741	1.06
	submitResultHash	41,815	0.0003387015	1.06

Table 5 presents the execution costs of the smart contract with varying numbers of workers and voters participating. As the number of workers increases, the cost per worker stabilizes around USD 4.50. The underlying reason for this is that although the total execution cost of the smart contract increases with the number of participants, certain costs, such as fixed function call fees, remain independent of the number of workers. Additionally, subsequent function calls generally incur lower costs than the first call, allowing the fixed costs to be distributed across the growing number of workers. This results in a smooth transition of the per-worker cost, which stabilizes at USD 4.50, a level that is economically acceptable. By leveraging blockchain and IPFS technologies, requesters can significantly reduce the additional costs typically associated with traditional intermediary services. Specifically, blockchain provides a decentralized trust mechanism that eliminates the need for costly centralized platforms, while its transparency and security further optimize the cost structure, enhancing the economic efficiency of the system.

Table 5. Total gas consumption for different numbers of participants.

No. of Workers	No. of Voter	Gas Used	ETH Value	Cost (USD)	Per Worker Cost (USD)
1	1	338,987	0.002746	8.572261	8.572261
5	1	1,032,231	0.008361	26.102930	5.220586
10	2	1,940,647	0.015719	49.074841	4.907484
20	4	3,757,479	0.030436	95.018663	4.750933
50	5	8,998,670	0.072889	227.557251	4.551145
100	10	17,873,525	0.144776	451.983484	4.519835

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 USD 16.5 when the number of bounding boxes hits 30. In contrast, off-chain storage costs remain relatively stable, fluctuating only slightly between USD 0.61 and USD 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 is stored on-chain. The CID is relatively short and fixed in length, and therefore, gas consumption remains low, regardless of data volume. This approach significantly alleviates on-chain storage demands and reduces storage-related gas costs, enhancing cost-effectiveness, particularly when handling large-scale data.

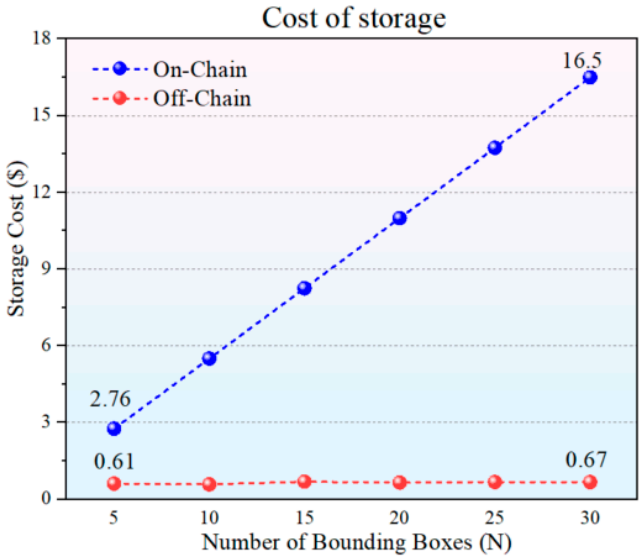


Figure 5. Comparison of on-chain and off-chain storage costs.

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 USD 174.71 when the count reaches 35. Conversely, the off-chain computation cost remains stable at USD 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 constant regardless of bounding box quantity. Consequently, the off-chain computation method effectively conserves on-chain computational resources, enhancing the scalability and cost-efficiency of CrowdBA.

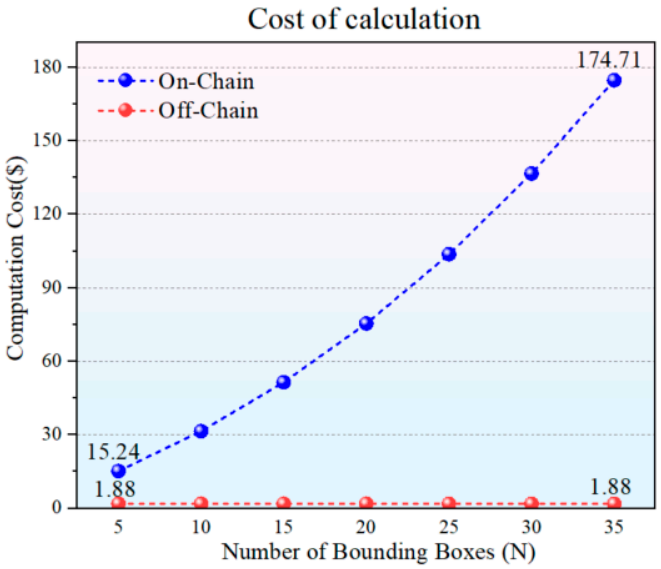


Figure 6. Comparison of on-chain and off-chain computation costs.

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. IPFS Data Takes Time to Upload and Download

To evaluate the performance of the IPFS network, this study analyzes the performance of uploading and downloading data files from the IPFS network. The test files ranged in size from 100 KB to 2000 KB, with the number of bounding boxes varying between 2158 and 203,639. A total of 100 independent experiments were conducted, during which the time taken for data uploads and downloads was recorded. The experimental results are presented in Table 6. It is evident that as the file size increases, the upload and download times on IPFS also increase. Since blockchain transactions rely on IPFS to complete their operations, larger files lead to longer processing times. However, in the context of bounding box data annotation, the stored files are generally small, consisting mostly of bounding box data. Therefore, the latency of IPFS uploads and downloads has a minimal impact on the overall architecture performance.

Table 6. IPFS upload and download time.

File Size/kb	Bounding Number	Upload Time/ms	Download Time/ms
100	2158	673.97	633.57
200	4357	629.77	650.35
500	10,794	616.06	706.80
1000	21,587	560.21	686.03
2000	43,199	715.11	675.71

5.5. 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 2369 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 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.

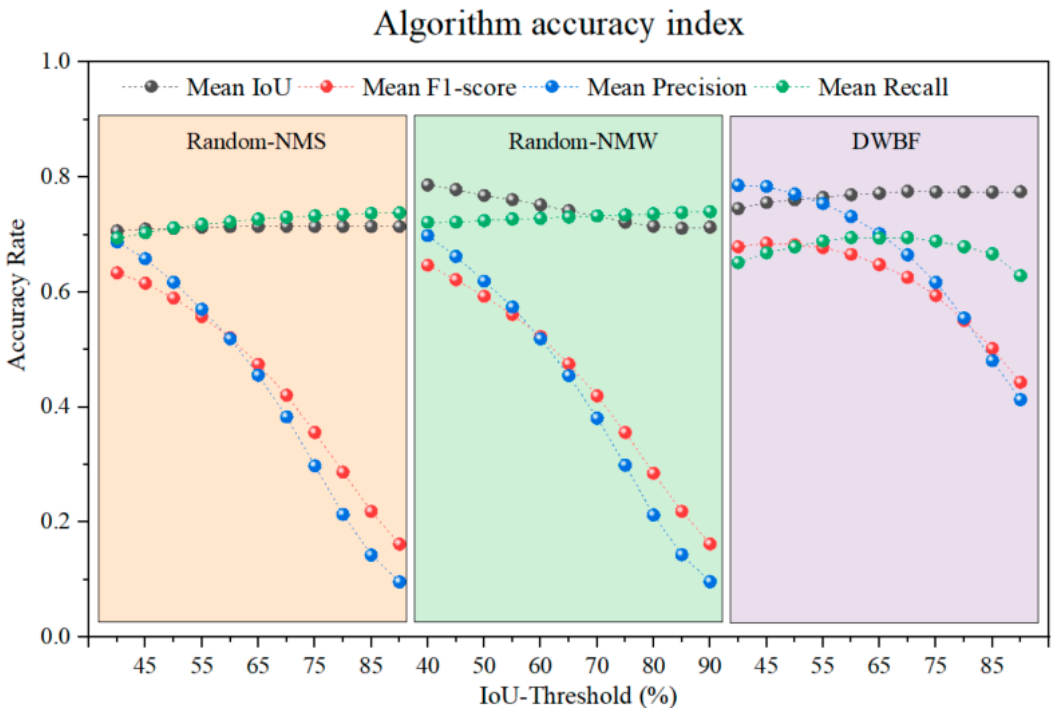


Figure 7. Comparison of bounding box fusion algorithms.

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 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. In real-world applications, CrowdBA can be widely utilized in scenarios such as pedestrian and vehicle detection in autonomous driving, tumor or organ localization in medical image analysis, and object grasping in robot vision. These fields rely on large amounts of precise bounding box annotation data to achieve accurate object recognition and localization. Due to its scalability and cost-effectiveness, CrowdBA is poised to become an ideal solution when there is insufficient labeled data for large-scale machine learning and AI model training. Future research may explore blockchain consensus algorithms [27–29] or Layer 2 scaling solutions [30–32] to further reduce on-chain operational costs and enhance the scalability of the CrowdBA architecture. Additionally, extending the annotation model to include speech labeling [33], semantic segmentation [34], and sentiment labeling [35] could provide more efficient and reliable data support for AI and machine learning applications.

Author Contributions: R.G. contributed to formal analysis, writing—review and editing, supervision, project management, and funding acquisition; S.L. contributed to writing—original draft preparation, methodology, software, and validation of the proposed scheme; J.Z. contributed to writing—review and editing. All authors have read and agreed to the published 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 experimental codes and datasets are available at https://github.com/Aurevior636/CrowdBA.git (accessed on 13 January 2025).

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

References

1. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep learning for generic object detection: A survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]

2. Kaur, J.; Singh, W. Tools, techniques, datasets and application areas for object detection in an image: A review. *Multimed. Tools Appl.* **2022**, *81*, 38297–38351. [CrossRef] [PubMed]
3. Whang, S.E.; Roh, Y.; Song, H.; Lee, J.G. Data collection and quality challenges in deep learning: A data-centric AI perspective. *VLDB J.* **2023**, *32*, 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. *Appl. Sci.* **2023**, *13*, 7082. [CrossRef]
5. Wylde, V.; Rawindaran, N.; Lawrence, J.; Balasubramanian, R.; Prakash, E.; Jayal, A.; Khan, I.; Hewage, C.; Platts, J. Cybersecurity, data privacy and blockchain: A review. *SN Comput. Sci.* **2022**, *3*, 127. [CrossRef] [PubMed]
6. Xia, H.; McKernan, B. Privacy in Crowdsourcing: A Review of the Threats and Challenges. *Comput. Support. Coop. Work (CSCW)* **2020**, *29*, 263. [CrossRef]
7. Jia, L.; Shao, B.; Yang, C.; Bian, G. A Review of Research on Information Traceability Based on Blockchain Technology. *Electronics* **2024**, *13*, 4140. [CrossRef]
8. Alsadhan, A.; Alhogail, A.; Alsalamah, H. Blockchain-Based Privacy Preservation for the Internet of Medical Things: A Literature Review. *Electronics* **2024**, *13*, 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. *Electronics* **2024**, *13*, 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. *Commun. Monogr.* **2018**, *85*, 140–156. [CrossRef]
12. CrowdSFL: A secure crowd computing framework based on blockchain and federated learning. *Electronics* **2020**, *9*, 773. [CrossRef]
13. Yang, S.; Zhang, Y.; Cui, L.; Deng, B.; Chen, T.; Dong, Q. A Web 3.0-Based Trading Platform for Data Annotation Service With Optimal Pricing. *IEEE Trans. Netw. Sci. Eng.* **2023**, *11*, 4032–4044. [CrossRef]
14. Witowski, J.; Choi, J.; Jeon, S.; Kim, D.; Chung, J.; Conklin, J.; Longo, M.G.F.; Succi, M.D.; Do, S. MarkIt: A collaborative artificial intelligence annotation platform leveraging blockchain for medical imaging research. *Blockchain Healthc. Today* **2021**, *4*. [CrossRef]
15. Zhu, X.R.; Wu, H.H.; Hu, W. FactChain: A Blockchain-based Crowdsourcing Knowledge Fusion System. *J. Softw.* **2022**, *33*, 3546–3564. (In Chinese) [CrossRef]
16. Li, M.; Weng, J.; Yang, A.; Lu, W.; Zhang, Y.; Hou, L.; Liu, J.-N.; Xiang, Y.; Deng, R.H. CrowdBC: A blockchain-based decentralized framework for crowdsourcing. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *30*, 1251–1266. [CrossRef]
17. Han, X.; Liu, Y. Research on the consensus mechanisms of blockchain technology. *Netinfo Secur.* **2017**, *17*, 147–152. [CrossRef]
18. Xu, J.; Wang, C.; Jia, X. A survey of blockchain consensus protocols. *ACM Comput. Surv.* **2023**, *55*, 1–35. [CrossRef]
19. de Ocáriz Borde, H.S. *An Overview of Trees In Blockchain Technology: Merkle Trees and Merkle Patricia Tries*; University of Cambridge: Cambridge, UK, 2022.
20. Taherdoost, H. Smart contracts in blockchain technology: A critical review. *Information* **2023**, *14*, 117. [CrossRef]
21. Khan, S.N.; Loukil, F.; Ghedira-Guegan, C.; Benkhelifa, E.; Bani-Hani, A. Blockchain smart contracts: Applications, challenges, and future trends. *Peer Peer Netw. Appl.* **2021**, *14*, 2901. [CrossRef] [PubMed]
22. Cai, W.; Wang, Z.; Ernst, J.B.; Hong, Z.; Feng, C.; Leung, V.C.M. Decentralized applications: The blockchain-empowered software system. *IEEE Access* **2018**, *6*, 53019. [CrossRef]
23. Casino, F.; Dasaklis, T.K.; Patsakis, C. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telemat. Inform.* **2019**, *36*, 55. [CrossRef]
24. Liu, Z.; Luong, N.C.; Wang, W.; Niyato, D.; Wang, P.; Liang, Y.-C.; Kim, D.I. A survey on blockchain: A game theoretical perspective. *IEEE Access* **2019**, *7*, 47615. [CrossRef]
25. Xu, X.; Weber, I.; Staples, M.; Zhu, L.; Bosch, J.; Bass, L.; Pautasso, C.; Rimba, P. A taxonomy of blockchain-based systems for architecture design. In Proceedings of the 2017 IEEE international conference on software architecture (ICSA), Gothenburg, Sweden, 3–7 April 2017; p. 243. [CrossRef]
26. Wang, L.; Yang, Y.; Wang, Y. Do higher incentives lead to better performance?-An exploratory study on software crowdsourcing. In Proceedings of the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Porto de Galinhas, Brazil, 19–20 September 2019. [CrossRef]
27. Wang, W.; Hoang, D.T.; Hu, P.; Xiong, Z.; Niyato, D.; Wang, P.; Wen, Y.; Kim, D.I. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access* **2019**, *7*, 22328. [CrossRef]
28. Lashkari, B.; Musilek, P. A comprehensive review of blockchain consensus mechanisms. *IEEE Access* **2021**, *9*, 43620. [CrossRef]
29. Ferdous, M.S.; Chowdhury, M.J.M.; Hoque, M.A.; Colman, A. Blockchain consensus algorithms: A survey. *arXiv* **2020**, arXiv:2001.07091. [CrossRef]

30. Gudgeon, L.; Moreno-Sanchez, P.; Roos, S.; McCorry, P.; Gervais, A. Sok: Layer-two blockchain protocols. In Proceedings of the Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, 10–14 February 2020; Springer: Berlin/Heidelberg, Germany, 2020. [CrossRef]
31. Gangwal, A.; Gangavalli, H.R.; Thirupathi, A. A survey of Layer-two blockchain protocols. *J. Netw. Comput. Appl.* **2023**, *209*, 103539. [CrossRef]
32. Thibault, L.T.; Sarry, T.; Hafid, A.S. Blockchain scaling using rollups: A comprehensive survey. *IEEE Access* **2022**, *10*, 93039. [CrossRef]
33. Malik, M.; Malik, M.K.; Mehmood, K.; Makhdoom, I. Automatic speech recognition: A survey. *Multimed. Tools Appl.* **2021**, *80*, 9411. [CrossRef]
34. Yu, H.; Yang, Z.; Tan, L.; Wang, Y.; Sun, W.; Sun, M.; Tang, Y. Methods and datasets on semantic segmentation: A review. *Neurocomputing* **2018**, *304*, 82. [CrossRef]
35. Bostan LA, M.; Klinger, R. An analysis of annotated corpora for emotion classification in text. In Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018; pp. 2104–2119.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.