NMFT: A Copyrighted Data Trading Protocol based on NFT and AI-powered Merkle Feature Tree

Dongming Zhang, Lei Xie and Yu Tao

***Abstract*—With the rapid growth of blockchain-based Non-Fungible Tokens (NFTs), data trading has evolved to incorporate NFTs for ownership verification. However, the NFT ecosystem faces significant challenges in copyright protection, particularly when malicious buyers slightly modify the purchased data and re-mint it as a new NFT, infringing upon the original owner's rights. In this paper, we propose a copyright-preserving data trading protocol to address this challenge. First, we introduce the Merkle Feature Tree (MFT), an enhanced version of the traditional Merkle Tree that incorporates an AI-powered feature layer above the data layer. Second, we design a copyright challenge phase during the trading process, which recognizes the data owner with highly similar feature vectors and earlier on-chain timestamp as the legitimate owner. Furthermore, to achieve efficient and low-gas feature vector similarity computation on blockchain, we employ Locality-Sensitive Hashing (LSH) to compress high-dimensional floating-point feature vectors into single uint256 integers. Experiments with multiple image and text feature extraction models demonstrate that LSH effectively preserves the similarity between highly similar feature vectors before and after compression, thus supporting similarity-based copyright challenges. Experimental results on the Ethereum Sepolia testnet demonstrate NMFT's scalability with sublinear growth in gas consumption while maintaining stable latency.**

***Index Terms*—Copyright protection, non-fungible token (NFT), Merkle tree, AI, blockchain, data trading**

I. Introduction

THE proliferation of substantial volumes of data, driven by the rapid development of the information technology, has significantly boosted various sectors, including government [1], enterprises [2] and research institutions [3], enhancing their efficiency and innovation capabilities. However, relying exclusively on one's own data is insufficient to exploit its potential value, creating an urgent demand for data exchange. Consequently, an abundance of data trading marketplaces such as Datacoup [4] and Qlik [5] have rapidly emerged. Nonetheless, such centralized marketplaces exhibit several drawbacks. The complexity of the trading process leads to prolonged trading time, and malicious behaviors within the marketplace itself may result in data breaches. Additionally, external attacks on the marketplace can cause a single point of failure, rendering services unavailable.

Blockchain, as a decentralized distributed system [6], [7], addresses the aforementioned weaknesses in the traditional data trading marketplace. On the one hand, it leverages the peer-to-peer nature of blockchain to enable direct data exchange between two parties, bypassing cumbersome procedures and reducing the risk of third-party malicious behaviors. Smart contracts [8] further automate data transactions by executing predefined rules without the need for human intervention. On the other hand, the maintenance of a distributed ledger, along with partially malicious-tolerant consensus mechanisms [9], [10], and the design of hash links between adjacent blocks, make malicious attacks on blockchain prohibitively costly.

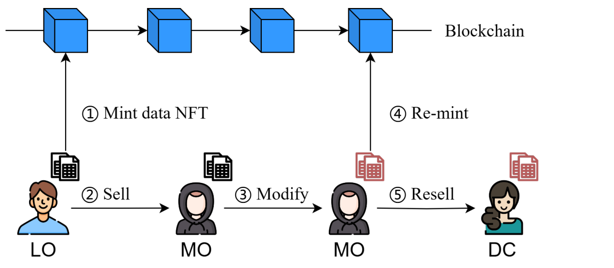


Fig. 1: The copyright protection challenge in NFT-based data trading, where a legitimate owner (LO) sells data to a misappropriator (MO), who then makes minimal modifications and re-mints it as a new NFT before reselling to a normal data consumer (DC), infringing the original owner's copyright.

In recent years, the Non-fungible token (NFT) [11] on the blockchain has garnered significant attention. Its capacity to provide verifiable proof of ownership and a traceable ownership history makes it highly suitable for data exchange scenarios. Several studies, including Song et al. [12] and Ranjbar et al. [13] have explored the use of data as an NFT to fully leverage the aforementioned characteristics. In addition, a number of decentralized data trading platforms emerged [14]–[16], allowing customers to mint and sell data NFTs.

Although blockchain and its associated NFTs have addressed some of the challenges in traditional data trading, one significant challenge remains unresolved: *copyright protection*. Specifically, the concern arises when a misappropriator (MO), after acquiring data from a legitimate owner (LO), make minimal modifications or leave it unaltered, and then re-mint it as a novel NFT (known as copymint) for reselling to a normal data consumer (DC) as shown in Fig. 1. The minimal modifications may contain color adjustment, noise addition, filtering for image data and character replacement, text rearrangement, synonym replacement for text data. Such actions could not only sever the NFT traceability, but also infringe upon the intellectual property rights of LO.

Previous studies that have utilized data as an NFT, such

Dongming Zhang and Lei Xie are with Zhejiang Lab, Hangzhou 311121, China. (e-mail: zhangdongming@zhejianglab.com; xielei@zhejianglab.com).

Yu Tao is with College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China. (e-mail: yu_tao@nuaa.edu.cn).

Corresponding author: Dongming Zhang (e-mail: zhangdongming@zhejianglab.com).

as those referenced in [12], [13], have not considered the copyright protection problem. Specifically, the work presented in [12] employed NFT to provide dynamic data traceability, privacy and exchange fairness. It is noteworthy that they designed an efficient generic transformation protocol to verify all transformations with zero-knowledge proofs. However, this system is vulnerable to disruption if a malicious buyer were to re-mint a data as a new NFT after obtaining the data without acquiring the NFT ownership, thereby severing the transformation relationship. Similarly, the research in [13] utilized digital watermarking alongside NFT to ensure data traceability against malicious resales. However, their approach fails to prevent a malicious buyer from listing the data as a new NFT, which bypasses any linkage to the previous owner. In summary, this action not only undermines the original creator's copyright but also disrupts the intended traceability and ownership verification mechanisms.

In the industry, there are many copymint cases in nowadays NFT marketplace. For instance, Bored Ape Yacht Club (BAYC) has a collection of 10,000 original NFTs and more than 4 million counterfeit NFTs [17]. Opensea [18], currently the largest NFT marketplace, self-reported that more than 80% of its NFTs minted for free were unoriginal or fake [19].

However, the current NFT market and its underlying blockchain infrastructure face significant challenges in detecting unauthorized data alterations and preventing the subsequent re-minting of data as new NFTs. According to OpenSea's copymint policy [20], the process of verifying potential copymints involves both image detection technology and human review. This approach, while necessary to combat infringement, reintroduces an element of centralization into the NFT trading lifecycle, which contradicts the decentralized ethos of blockchain technology.

Confronted with the aforementioned copyright protection challenge, we propose a novel data trading protocol named NMFT based on NFT and Merkle Feature Tree (MFT). This protocol offers a decentralized solution to the copyright challenge in the current data trading landscape. Specifically, a reconstructed Merkle Tree, called the MFT, is designed to capture the features of the underlying data and embed them into the Merkle root. A verifiable random-selection mechanism is introduced to ensure the randomness of the chosen data and feature samples, thereby preventing data owner fraud. The randomly selected samples are then subjected to a copyright challenge to verify the legality of NFT ownership by smart contract, using feature similarity. This approach aligns more closely with Web3 decentralized principles compared to current solutions that rely on third-party and manual oversight for validation. The main contributions of this paper are summarized as follows:

- We propose NMFT, a novel data trading protocol that integrates NFT with MFT to address copyright protection in blockchain-based data exchanges. By extending the traditional exchange fairness to include copyright fairness and ownership transfer fairness, NMFT provides a comprehensive solution for secure and fair data trading.
- We introduce the MFT, an enhanced version of the traditional Merkle Tree that incorporates AI-extracted features above the data layer. This structure embeds distinctive data features within the Merkle root, which is immutably recorded on the blockchain, enabling efficient copyright verification even when data has been slightly modified.
- We implement an efficient on-chain similarity computation scheme that compresses AI-extracted high-dimensional feature vectors into single uint256 integers using Locality-Sensitive Hashing (LSH). This makes copyright verification practically feasible on blockchain while maintaining its effectiveness.

TABLE I: COMPARISON WITH EXISTING WORKS.

Scheme	Data Traceability[2]	Exchange Fairness	Copyright Protection[1]	Data Privacy
[12]	Native	✓	None	✓
[13]	Native	✓	Reactive	×
[14]	Native	✓	None	✓
[15]	Native	✓	None	✓
[21]	None	✓	None	✓
[22]	None	✓	None	✓
[23]	None	✓	None	✓
[24]	None	✓	None	✓
[25]	Auxiliary	✓	Reactive	✓
[26]	Auxiliary	✓	Proactive	✓
[27]	Auxiliary	✓	Reactive	×
[28]	Auxiliary	✓	Reactive	✓
[29]	Auxiliary	✓	Proactive	✓
Our NMFT	**Native**	✓	**Proactive**	✓

[1] For "Copyright Protection": None means no specific protection mechanism; Reactive indicates post-hoc detection and punishment of infringement; Proactive provides prevention of infringement before it occurs.

[2] For "Data Traceability": None means only recording transactions on blockchain without ownership verification; Auxiliary represents using additional auxiliary techniques like watermarking, perceptual hashing, or cryptographic proofs to assist ownership verification, but lacks native uniqueness guarantees; Native indicates native support for unique ownership and transfer verification through NFTs.

The rest of the paper is organized as follows. Section II reviews the related work. Section III presents the preliminaries. Section IV shows the system model, the threat model and the design goals. Section V describes the detailed design of NMFT. Section VI gives the system analysis. Section VII shows the experimental results and the last section summarizes the paper.

II. Related Work

In this section, we review the existing works on data trading with respect to data NFTs, exchange fairness and copyright protection, with a focus on studies that incorporate blockchain technology. The comparison between our schemes and the existing works is shown in Table I.

A. Data NFTs

In recent years, both the academic and industrial sectors have explored and applied the concept of digitizing data assets as NFTs, taking advantage of their unique traceability feature.

Within the academic community, a number of studies have been introduced to exploit the traceability of NFTs.

For example, Song et al. [12] focused on tracking data transformation processes in machine learning scenarios. Their system, ZKDET, enables the tracing of data transformations (e.g., aggregation, partitioning) and verifies the correctness of these transformations using zero-knowledge proofs. This allows data owners to monetize not only the raw data but also its derived products through model training and data mining. Furthermore, to protect data privacy while ensuring verifiable transformations, they proposed a generic transformation protocol that employs zero-knowledge proofs. However, ZKDET primarily focuses on verifying data transformations in machine learning applications and does not consider copyright protection in direct data trading. In another study, Ranjbar et al. [13] proposed a watermark-based NFT framework that enables royalty payments to original data owners in subsequent resales. They provide a reactive protection mechanism where original owners can report copyright violations by verifying the embedded watermarks. Nevertheless, their system relies on a trusted third party to embed robust watermarks into the data prior to NFT minting, which requires exposing the original data to the third party and potentially compromises data privacy. Neither of these systems provides proactive mechanisms to prevent unauthorized NFT trading, which could lead to parallel markets of duplicated data NFTs and disrupt the proper valuation of data assets.

In the realm of industry, different approaches to NFT-based data ownership have emerged. Ocean Protocol [14] is a decentralized data exchange protocol that incentivizes the publishing of data for use in the training of AI models. Data assets are treated as data NFTs and exchanged by its own datatokens which live on multiple blockchains including Ethereum, Polygon, BNB Chain, etc. Additionally, a tool called Compute-to-Data within Ocean Protocol is provided to share or monetize one's data without compromising privacy. Nonetheless, there is an absence of explicit evidence to suggest that Ocean Protocol has implemented robust strategies to mitigate the occurrence of copymint activities.

Itheum [15] is another data exchange protocol built on blockchain that transforms the data asset ownership landscape. In the process of minting a data NFT on Itheum, data owners are granted the capability to stipulate the quantity of copies, denoting the total number of identical replicas of the data NFT that can be minted. However, this mechanism does not extend its efficacy to preclude instances of copymint behaviors that involve minimal modifications to the original asset.

B. Exchange Fairness

Exchange fairness is a basic requirement of a data trading protocol. Researchers have proposed numerous solutions to tackle this issue [21]–[24]. Dziembowski et al. [21] proposed a protocol which allows fair sale of a witness where a judge smart contract verifies concise proofs of misbehavior. Delgado-Segura et al. [22] provided a fair protocol for data trading based on the private key locked transaction on Bitcoin network. Chenli et al. [23] introduced a novel atomic data exchange protocol named Fair2Trade, which utilizes a verifiable statement protocol based on the idea of Merkle Tree and hash chain structures, thus ensuring both exchange and distribution fairness. Abla et al. [24] proposed a data trading protocol to achieve fairness by combining the probabilistic approaches with fully homomorphic encryption.

C. Copyright Protection

Copyright protection is another critical aspect within data trading. Various strategies have been proposed to address this challenge, with digital watermarks and fingerprints being predominant solutions [25]–[27]. Sheng et al. [25] developed CPChain, a blockchain-based crowdsourcing data trading framework that combines digital fingerprints with homomorphic encryption. Although their system preserves data privacy during trading, the copyright protection mechanism relies on post-hoc detection through fingerprint tracing rather than proactive prevention. Xiang et al. [26] introduced five algorithms designed to proactively safeguard data copyright through watermark-based policies and smart contracts, effectively preventing unauthorized redistribution, combination, and leakage of data. However, both of [25] and [26] assumed that there exists robust watermark or fingerprint method. Wang et al. [27] focused on copyright protection within a medical image data trading platform. They utilized a U-Net-based [30] CNN [31] architecture to segment medical images into regions of interest and non-interest, embedding watermarks in the latter to safeguard copyright without compromising diagnostic integrity. However, their approach requires a trusted third party for watermark generation and embedding, which introduces privacy risks as the original data must be exposed to this third party.

However, the incorporation of digital watermarks or fingerprints for copyright protection can potentially compromise the integrity of the original data. Furthermore, there is a risk that malicious buyers may attempt to circumvent the protective measures by manipulating the data, thereby diminishing the effectiveness of the watermarks or fingerprints.

In addition to the aforementioned methods, alternative techniques have been explored for safeguarding copyrights. Wang et al. [28] proposed a protocol for copyright-preserving data trading, incorporating a novel perceptual hashing algorithm based on ResNet-18 [32]. Chen et al. [29] presented a blockchain-driven copyright protection scheme, which incorporates double-authentication-prevention signatures and non-interactive zero-knowledge proofs to proactively detect and deter illegal copyright transfers.

III. Preliminaries

A. Merkle Tree and Merkle Proof

The Merkle Tree [33] is a binary tree where each leaf node contains the hash of a data block, and each non-leaf node contains the hash of its children. This process continues recursively until reaching the Merkle root.

The Merkle proof efficiently verifies whether a specific data block is included in a Merkle Tree. As shown in Fig. 2, to verify data block B, one only needs h_A and $H(h_C|h_D)$ (shaded in grey). The verification involves hashing B and iteratively combining with these nodes to compare with the root.

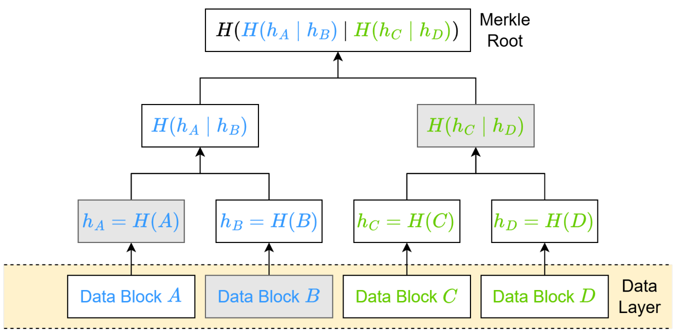


Fig. 2: The structure of Merkle Tree where $H(\cdot)$ denotes a public one-way hash function and "|" represents the concatenation between two hash values.

B. Merkle Feature Tree

We propose the Merkle Feature Tree (MFT) that adds a feature layer above the data layer as shown in Fig. 3. The feature layer extracts vectors using models like MPNet [34], MiniLM [35] for text, and DINOv2 [36], Swin Transformer [37] for images. These vectors are then hashed following standard Merkle Tree procedures.

The integration of a feature layer introduces a critical advantage in maintaining data ownership. When MO presents a feature vector from modified data, LO can counter with a similar vector registered earlier, along with its Merkle proof. This proves legitimate ownership by showing precedence on the blockchain. All submitted vectors must have valid Merkle proofs linking to recorded Merkle roots, preventing arbitrary submissions.

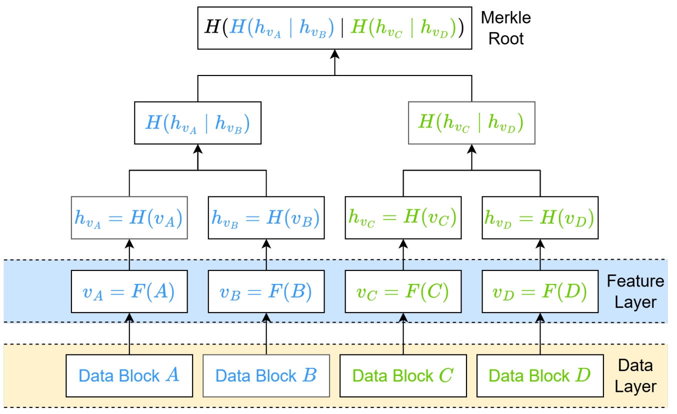


Fig. 3: The structure of Merkle Feature Tree (MFT) where $H(\cdot)$ and $F(\cdot)$ denote a public one-way hash function and feature extraction method respectively, and "|" represents the concatenation between two hash values.

C. Non-Fungible Tokens

Non-fungible Tokens (NFTs) are a type of digital asset created via smart contracts on the blockchain. Different from fungible tokens like BTC in Bitcoin [6] or ETH in Ethereum [7], which are interchangeable and divisible, each NFT is distinct and cannot be exchanged with another NFT on a one-to-one basis. The uniqueness is ensured through a distinct identifier, known as $tokenID$ generated by the smart contract. Due to the high storage costs associated with the blockchain, a hybrid approach of on-chain and off-chain method is typically used. Specifically, the actual digital asset is usually stored in off-chain distributed storage systems like IPFS [38], while the corresponding storage address and some metadata are recorded on the blockchain.

D. Hash Chain

A hash chain [39] is constructed by iteratively applying a hash function $H(\cdot)$ on a random seed s:

$$\{s, H(s), H^2(s), H^3(s), ..., H^{n-1}(s), H^n(s)\},$$

where $H^{i+1}(s) = H(H^i(s))$, and $H^n(s)$ is called the tip. Given the preimage resistance of $H(\cdot)$, one with computational limitations cannot derive previous values from $H^i(s)$, but can easily compute subsequent values to the tip.

IV. Model and Design

A. System Model

In the proposed data trading protocol, three key entities are identified: the data owner, the data buyer, and the data trading smart contract denoted by SC. The data owner is further divided into two distinct sub-entities: *the legitimate data owner* represented by LO and *the data misappropriator* abbreviated as MO. LO possesses valuable data assets with the intention to monetize them through sale. Conversely, MO engages in potentially malicious behaviors by purchasing data from other data owners without acquiring the NFT ownership. Subsequently, they may re-mint this data as a new data NFT, possibly with minor data modifications, in an attempt to profit illegally. Meanwhile, the data buyer is seeking the desired data from the data owner. SC serves as a trusted third party, facilitating the connection between the data owner and the data buyer to fulfill data trading. Importantly, it has the capability to distinguish between LO and MO. The proposed data trading protocol includes the following five phases:

- **P1: Setup.** In this phase, the administrator deploys a data trading smart contract named SC.
- **P2: Mint data NFT.** The data owner mints his data NFT via the deployed SC with some essential meta data recorded on the blockchain.
- **P3: Request data.** The data buyer sends a data request to SC in order to acquire the desired data or the data NFT in this phase.
- **P4: Challenge.** In this challenge phase, the data buyer issues a challenge to the owner to verify both the validity of the data and the legitimacy of the copyright. The owner must respond with the corresponding data blocks and feature vectors for verification by the buyer and potentially other owners acting as challengers.
- **P5: Batch payment.** The data owner delivers the data to the buyer in batches, and upon each round of confirmation for data validity, the buyer transfers a verifiable payword to the owner. Once the entire data transfer is complete, the last payword and the completed batch number are submitted to SC to process the payment.

B. Threat Model

In all five phases of the data trading protocol, there is a possibility of malicious behavior among the three key entities. As previously mentioned, the data trading smart contract SC is considered a trusted entity, as it is deployed on a secure blockchain. Therefore, our focus is on the potential malicious behaviors that may occur between the data owner and the data buyer.

- **Malicious data owner:** The malicious behaviors can consist of the following types: 1) as a copyright infringer: acquiring data from legitimate owners and reselling the data to gain profit; responding with incorrect challenge messages during the copyright challenge phase; transferring incorrect data or mixing fake data with real ones during batch data transfer; 2) as a challenger: attempting to falsely claim ownership during copyright challenges without possessing legitimate data.
- **Malicious data buyer:** The malicious behaviors can include: 1) refusing to deliver paywords or sending invalid paywords after receiving data batches; 2) trying to bypass the legitimate owner by submitting the payword directly to the smart contract; 3) attempting to obtain NFT ownership without completing all required payments.

C. Design Goals

Based on exchange fairness requirements and NFT incorporation, we propose enhanced exchange fairness and data privacy as key design goals.

Definition 1: (Enhanced exchange fairness). *A data trading protocol achieves enhanced exchange fairness if: 1) basic exchange fairness: buyer accesses data only after payment, owner gets paid only after data transfer; 2) copyright fairness: owner verified as* LO *before payment; 3) ownership transfer fairness: buyer acquires NFT ownership only after full payment including transfer fee, owner receives complete payment only after NFT ownership transfer.*

Definition 2: (Data privacy). *A data trading protocol achieves data privacy if: 1) data known only to owner before exchange; 2) minimal data exposure for validation during trading; 3) buyer accesses complete/partial data post-exchange based on payment extent.*

V. System Design

A. Data Trading Type and Transfer Type

Data treated as an NFT offers the benefit of establishing clear ownership and tracking its history. We summarize two types of data trading:

- **Trade data itself (TDI):** For buyers seeking to access a subset of the data without taking over the NFT ownership. The data owner retains the NFT and can continue trading with other buyers.
- **Trade data & ownership (TDO):** For buyers interested in both the data and NFT ownership. Upon completion, the original owner can no longer sell the data, as the buyer becomes the new owner.

These two trading types lead to different pricing structures: TDI solely accounts for the price of the partial data, whereas TDO encompasses the price of the whole data plus an additional fee for NFT ownership transfer.

For data transfer, we advocate batch transfer in NMFT, utilizing a micropayment protocol [40], [41] to mitigate gas fees. This approach enables incremental verification, allowing buyers to assess each batch for quality and relevance.

B. Workflow of Protocol

Fig. 4 illustrates the comprehensive protocol design, which is structured into five distinct phases as follows.

Setup.

① **Admin deploy smart contract:** The administrator deploys a data trading smart contract on the blockchain with specific parameters initialized.

Mint data NFT.

② **Owner calculate MFT and hashes:** The data owner divides data D into n_d blocks $D = \{d_i\}$, where $i \in [1, n_d]$, and computes MFT with Merkle root r. The block size is standardized across all data owners (e.g., 100 lines of CSV or 10 images per block). The owner stores MFT locally for similarity computation and ownership proof, and calculates hash values $H_d = \{H(d_i)\}$ for all blocks.

③ **Owner mint data NFT:** The data owner mints NFT via SC, recording Merkle root, block count, data description, block price and NFT transfer fee in the NFT metadata. The generated $tokenID$ of the NFT will then be maintained by SC. Upon data update, the owner recalculates MFT and refreshes metadata with current r and n_d values.

Request data.

④ **Buyer request data:** The data buyer initiates a request via SC with trading parameters: data trading type $dtype$, number of data blocks per batch n_e, batch price p_b, desired batch number n_b, challenge size n_c, NFT transfer fee p_o and owner penalty amount p_p. After consensus, the owner can access this request on blockchain.

⑤ **Owner confirm request:** The data owner confirms all trading parameters ($dtype$, n_e, p_b, n_b, n_c, p_o, p_p) via SC.

⑥ **Buyer deposit money:** The data buyer deposits payment into SC: $n_b \cdot p_b$ for TDI or $n_b \cdot p_b + p_o$ for TDO.

⑦ **Owner deposit money:** The data owner deposits penalty amount p_p into SC as potential MO security.

⑧ **Transfer hash values:** The data owner transfers precalculated H_d to buyer through secure off-chain channel.

Challenge.

⑨ **Buyer initiate challenge:** The data buyer generates a random challenge list $L_c = \{j\}$ with length n_c. To reduce gas costs, the buyer sends L_c to the data owner through off-chain channel and initiates a challenge transaction via SC. This challenge consists of two parts: a *validity challenge* to verify data authenticity, and a *copyright challenge* to verify ownership legitimacy.

⑩ **Owner respond validity challenge:** After receiving L_c, the data owner extracts the corresponding challenge data

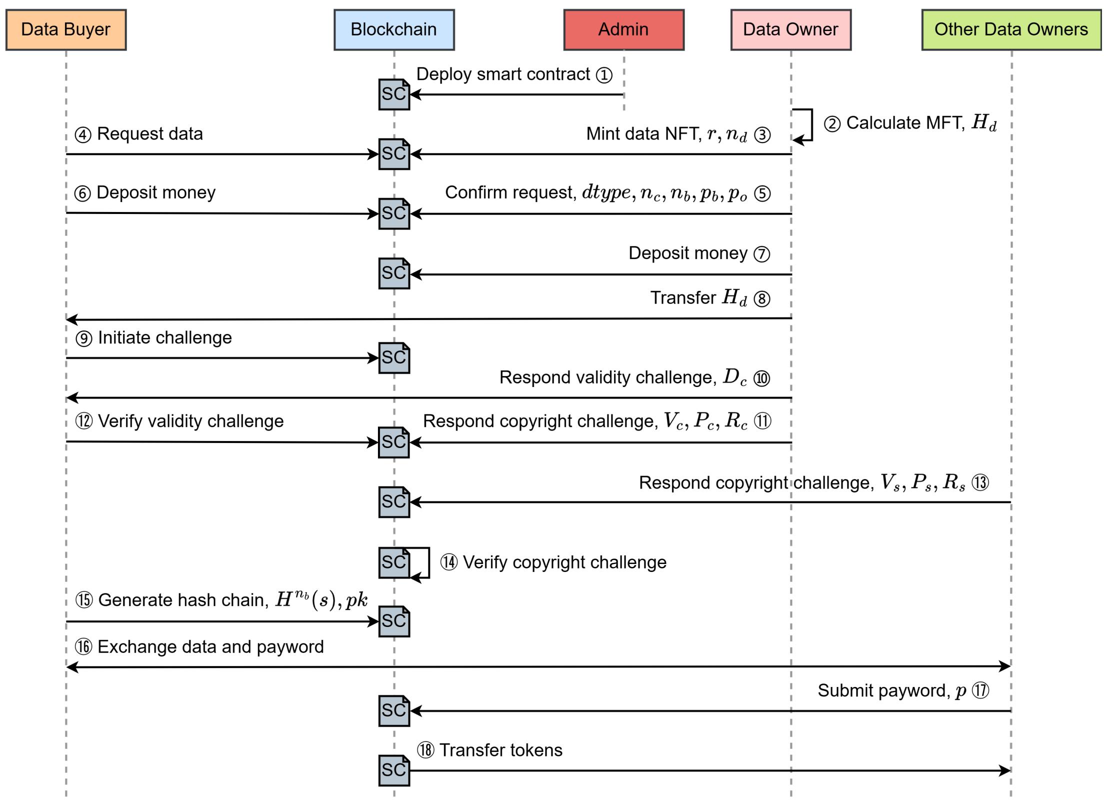

Data Buyer
Blockchain
Admin
Data Owner
Other Data Owners
Deploy smart contract ①
SC
② Calculate MFT, H_d
④ Request data
Mint data NFT, r, n_d ③
⑥ Deposit money
Confirm request, $dtype, n_c, n_b, p_b, p_o$ ⑤
Deposit money ⑦
Transfer H_d ⑧
⑨ Initiate challenge
Respond validity challenge, D_c ⑩
⑫ Verify validity challenge
Respond copyright challenge, V_c, P_c, R_c ⑪
Respond copyright challenge, V_s, P_s, R_s ⑬
⑭ Verify copyright challenge
⑮ Generate hash chain, $H^{n_b}(s), pk$
⑯ Exchange data and payword
Submit payword, p ⑰
⑱ Transfer tokens

Fig. 4: Workflow of NMFT, where the data owner (light red box) is exposed as MO and one challenger (light green box) succeeds as LO. Steps ② and ③ for challengers are identical to those of the data owner.

blocks $D_c = \{d_j\}$, where $d_j = D[j]$ and $j \in L_c$. The owner then transfers D_c to the buyer through a secure off-chain channel to minimize on-chain costs.

⑪ **Owner respond copyright challenge:** Using the challenge data D_c, the owner extracts feature vectors $V_c = \{v_j\}$, where $v_j = F(d_j)$ is computed using predefined feature extraction method F (optimized in Section VII-C). The owner then submits V_c, corresponding Merkle proofs P_c, Merkle roots R_c, and $tokenID$ to SC following Algorithm 1. To win the copyright challenge, the owner must use the earliest registered root r in R_c.

⑫ **Buyer verify validity challenge:** The buyer first verifies that both D_c and V_c have length n_c. Then computes hash values $H_c = \{H(d_j)\}$ and compares them with previously received $H_d[j]$ ($j \in L_c$) to verify data authenticity. The buyer also checks if D_c meets requirements $\mathcal{D}$ and recalculates feature vectors to verify V_c correctness. Algorithm 2 details this process, after which the buyer updates the bool variable $isDataOK$ in SC.

⑬ **Other owners respond copyright challenge:** Anyone interested in engaging in this copyright challenge is called the copyright challenger (denoted by CC), and must identify the related data NFT with a data description that matches their own data NFT. Subsequently, he/she obtains the challenge feature vectors V_c associated with the identified data NFT to proceed the following step of data similarity calculation. For any feature vector $v \in V_c$, CC computes the cosine similarity (optimized in Section VII-C) between v and his local feature vectors

Algorithm 1 Copyright Challenge Response

```
▷ Data Owner
Input: $D_c$, $tokenID$
 1: initialize $V_c \leftarrow [\,]$
 2: for $d$ in $D_c$ do
 3:     $v \leftarrow F(d)$
 4:     append $v$ to $V_c$
 5: $P_c \leftarrow$ GetMerkleProofs$(V_c, tokenID)$
 6: $R_c \leftarrow$ GetMerkleRoots$(V_c, tokenID)$
 7: SendToSC$(V_c, P_c, R_c, tokenID)$
▷ Copyright Challenger CC
Input: $V_c$, $V_l$, $tokenID_s$ and $\eta$
 1: Initialize $V_s \leftarrow [\,]$
 2: for $v$ in $V_c$ do
 3:     $v_s \leftarrow$ QueryMostSimilarVec$(V_l, v)$
 4:     $c \leftarrow$ CalcSimilarity$(v_s, v)$
 5:     if $c < \eta$ then
 6:         break
 7:     else
 8:         append $v_s$ to $V_s$
 9: if $len(V_s) == len(V_c)$ then
10:     $P_s \leftarrow$ GetMerkleProofs$(V_s, tokenID_s)$
11:     $R_s \leftarrow$ GetMerkleRoots$(V_s, tokenID_s)$
12:     SendToSC$(V_s, P_s, R_s, tokenID_s)$
```

Algorithm 2 Validity Challenge Verification

```
▷ Data Buyer
Input: $\mathcal{D}$, $H_d$, $n_c$, $D_c$, $L_c$ and $V_c$
Output: $isDataOK$
1: Initialize $isDataOK \leftarrow$ True
2: if $(n_c \neq len(D_c))$ or $(n_c \neq len(V_c))$ then
3:     $isDataOK \leftarrow$ False
4:     return $isDataOK$
5: for $i = 1$ to $n_c$ do
6:     $d \leftarrow D_c[i]$
7:     $j \leftarrow L_c[i]$
8:     if $H(d) \neq H_d[j]$ then
9:         $isDataOK \leftarrow$ False
10:        return $isDataOK$
11: for $i = 1$ to $n_c$ do
12:     $d \leftarrow D_c[i]$
13:     if $d \notin \mathcal{D}$ then
14:         $isDataOK \leftarrow$ False
15:         return $isDataOK$
16: for $i = 1$ to $n_c$ do
17:     $d \leftarrow D_c[i]$
18:     $v \leftarrow V_c[i]$
19:     if $F(d) \neq v$ then
20:         $isDataOK \leftarrow$ False
21:         return $isDataOK$
22: return $isDataOK$
```

V_l, which were previously stored locally when calculating the MFT. This comparison is conducted to ascertain whether a highly similar vector v_s exists, defined by a similarity that exceeds a predetermined threshold η. This indicates that the underlying data blocks are close to each other. Finally, if CC successfully identifies all the most similar vectors denoted by V_s, he/she then sends V_s, corresponding Merkle proofs P_s, Merkle roots R_s and $tokenID_s$ to SC as described in Algorithm 1. Here we use a different subscript "s" to denote the values of CCs. To succeed in the copyright challenge, CC also needs to use the earliest registered r in R_s.

⑭ **SC verify copyright challenge:** The copyright challenge verification process consists of three main stages: *owner verification*, *challenger verification*, and *final resolution* as shown in Algorithm 3.

Owner verification: Upon reception of the challenge response from the data owner, SC initially validates whether the dimensions of V_c, P_c and R_c match n_c. Subsequently, for each triplet $(v, p, r) \in (V_c, P_c, R_c)$, SC performs two verifications: 1) confirms the authenticity of Merkle root r within the NFT metadata using VerifyMerkleRoot$(r, tokenID)$; 2) verifies the integrity of the correspondence between v, p, and r using VerifyMerkleProof$(H(v), p, r)$. If all triplets pass these checks, the status variable $isVecOK$ is set to True.

Challenger verification: When receiving responses from CCs, SC first checks the status variables $isVecOK$ and $isDataOK$. Subsequently, SC carries out the identical verification procedure that was applied during the owner verification stage. SC further performs three steps: 1) computes similarities between vectors in V_c and V_s; 2) verifies if all similarities exceed threshold η; 3) for multiple qualifying CCs, selects the winner based on the cumulative time difference Δ_t. This Δ_t is calculated as the sum of the individual time differences for each challenged data block. Each individual time difference is the duration between the Merkle root recording times of the CC's and the data owner's corresponding data blocks. The CC with the largest Δ_t is chosen as the winner.

Final resolution: The outcome is handled as follows: 1) if LO is found, the challenged owner's deposit p_p is distributed evenly between the buyer and LO; 2) LO's ID $ownerID_s$ is associated with $tokenID$ on the blockchain; 3) if no CCs respond or pass verification within period T, the challenged owner is deemed LO by default. Note that in subsequent phases,"data owner" refers to LO.

Batch payment.

⑮ **Buyer generate hash chain:** The data buyer generates a random string s locally and creates a hash chain $\mathcal{H}$, whose length is equal to $n_b + 1$. Then the buyer invokes the tip recording function within the smart contract by passing the tip $H^{n_b}(s)$ and $tokenID$ as arguments.

⑯ **Exchange data and payword:** The data owner transfers data in batches as mentioned in Sec. V-A via secure communication channels. For simplicity, we standardize the size of each batch to match the size of a single data block. Within the i-th round of data and payword exchange, the data buyer verifies both the hash and the content of the received batch data, as outlined in phase ⑫. Upon successful verification, the data buyer sends the previous hash $H^{n_b - i}(s)$ in the hash chain to the owner as a payword. The owner then confirms the validity of the payword by checking whether $H(H^{n_b - i}(s))$ matches $H^{n_b - i + 1}(s)$, which received at the $(i-1)$-th round. If the validation is successful, the following batch of data is then sent to the buyer.

⑰ **Owner submit payword:** After data transfer, the owner receives a hash chain $\mathcal{H}_r$, which may be a sub-chain of the original chain $\mathcal{H}$ if the buyer stops payment (e.g., when finding data unsatisfactory). Let p denote the first element of $\mathcal{H}_r$ (last received payword) and n_p denote its length (completed batch number). The owner submits $(p, n_p, tokenID, buyerID)$ to SC, and can continue submitting until the cumulative sum of n_p reaches the desired batch number n_b.

⑱ **SC transfer tokens:** Upon receiving the tuple of $(p, n_p, tokenID, ownerID, buyerID)$, SC checks whether the owner identified by $ownerID$ is LO of the NFT, which is identified by $tokenID$. Then SC iteratively computes the hash value of p for n_p times, and compare the result with the pre-stored tip on the blockchain. If the two values match, SC retrieves the historical sum of n_p (denoted as N_p). If $N_p + n_p$ and $dtype$ are equal to n_d and TDO respectively, SC transfers

Algorithm 3 Copyright Challenge Verification

```
▷ Smart Contract SC
Input: $V_c$, $P_c$, $R_c$, $tokenID$, $V_s$, $P_s$, $R_s$, $tokenID_s$,
       $isDataOK$, $n_c$, $\eta$, $T$, $p_p$, $ownerID$ and $buyerID$

Initialize $\Delta_t \leftarrow 0$
Initialize $isVecOK \leftarrow False$
Initialize $hasWinner \leftarrow False$
  ▽ Owner verification
upon receiving $(V_c, P_c, R_c, tokenID)$ do
  if $(n_c \neq len(V_c))$ or $(n_c \neq len(P_c))$ or $(n_c \neq len(R_c))$ then
    return
  for $i = 1$ to $n_c$ do
    $v, p, r \leftarrow V_c[i], P_c[i], R_c[i]$
    $vr \leftarrow$ VerifyMerkleRoot$(r, tokenID)$
    $vp \leftarrow$ VerifyMerkleProof$(H(v), p, r)$
    if $(vr ==$ False$)$ or $(vp ==$ False$)$ then
      return
  $isVecOK \leftarrow True$
  ▽ Challenger verifiction
upon receiving $(V_s, P_s, R_s, tokenID_s)$ do
  if $(isVecOK ==$ False$)$ or $(isDataOK ==$ False$)$ then
    return
  if $(n_c \neq len(V_s))$ or $(n_c \neq len(P_s))$ or $(n_c \neq len(R_s))$ then
    return
  for $i = 1$ to $n_c$ do
    $v', p', r' \leftarrow V_s[i], P_s[i], R_s[i]$
    $vr \leftarrow$ VerifyMerkleRoot$(r', tokenID_s)$
    $vp \leftarrow$ VerifyMerkleProof$(H(v'), p', r')$
    if $(vr ==$ False$)$ or $(vp ==$ False$)$ then
      return
  for $i = 1$ to $n_c$ do
    Initialize $\Delta'_t \leftarrow 0$
    $v \leftarrow V_c[i]$
    $v' \leftarrow V_s[i]$
    $c \leftarrow$ CalcSimilarity$(v, v')$
    $t \leftarrow$ GetMerkleRootTime$(r, tokenID)$
    $t' \leftarrow$ GetMerkleRootTime$(r', tokenID_s)$
    if $(c < \eta)$ or $(t' \geq t)$ then
      return
    $\Delta'_t \leftarrow \Delta'_t + t - t'$
  if $\Delta'_t > \Delta_t$ then
    $hasWinner \leftarrow True$
    $\Delta_t \leftarrow \Delta'_t$
    $ownerID_s \leftarrow$ GetOwner$(tokenID_s)$
  return
  ▽ Final resolution
upon elapsed time reaches $T$ do
  if $hasWinner == True$ then
    RecordWinner$(tokenID, ownerID_s)$
    TransferToken$(p_p/2, ownerID, ownerID_s)$
    TransferToken$(p_p/2, ownerID, buyerID)$
  else
    RecordWinner$(tokenID, ownerID)$
```

$n_p \cdot p_b + p_o$ to the owner's address and simultaneously changes the ownership of the NFT to the buyer. Otherwise, SC only sends $n_p \cdot p_b$ to the owner. The detailed process can be found in Algorithm 4.

Algorithm 4 Token Transfer

```
▷ Smart Contract SC
Input: $p$, $n_p$, $tokenID$, $ownerID$, $buyerID$,
       $p_b$, $p_o$, $n_d$, $dtype$
upon receiving $(p, n_p, tokenID, ownerID, buyerID)$ do
  if $ownerID \neq$ GetWinner$(tokenID)$ then
    return
  $t \leftarrow$ GetTip$(tokenID, ownerID, buyerID)$
  $t' \leftarrow p$
  for $i = 1$ to $n_p$ do
    $t' \leftarrow H(t')$
  if $t' \neq t$ then
    return
  $N_p \leftarrow$ GetNPSum$(tokenID, ownerID, buyerID)$
  if $N_p + n_p == n_d$ and $dtype ==$ TDO then
    $token \leftarrow n_p \cdot p_b + p_o$
    TransferToken$(token, buyerID, ownerID)$
    ChangeNFTOwner$(ownerID, buyerID, tokenID)$
  else
    $token \leftarrow n_p \cdot p_b$
    TransferToken$(token, buyerID, ownerID)$
  UpdateTip$(tokenID, buyerID, p)$
```

VI. System Analysis

In this section, we provide system analysis for NMFT to show how we achieve the design goals defined in Sec. IV-C.

A. Enhanced Exchange Fairness

Theorem 1: *The data buyer can only access the data after paying the corresponding fee to the owner, except that a small batch of data may be available for free.*

Proof: We prove Theorem 1 by demonstrating how the protocol prevents malicious behaviors where buyers attempt to acquire data without payment.

Firstly, when the buyer is malicious, he/she may not deliver the corresponding payword to the owner. Recall that the batch transfer is utilized in NMFT. Thereby, upon not receiving the payword, the owner can decide not to send the next batch of data. In such a case, only a batch of data is disclosed which is very small.

Secondly, the buyer might attempt to transfer a fake payword to the owner. However, this attack is prevented as the owner can verify the authenticity of the received payword by computing its hash value and comparing it with either the previously obtained payword or, in the case of the first batch transfer, the tip recorded on the blockchain.

Thirdly, since the buyer possesses the initial hash chain, they might attempt to submit paywords directly to SC before the owner does. However, this attack is prevented by SC as shown in Algorithm 4, which verifies that only LO (who has succeeded in the copyright challenge) can submit paywords for payment. □

Lemma 1: *Given a cryptographically secure hash function, it is computationally infeasible for anyone with polynomial-time capabilities to forge a valid (p, n_p)-pair that passes the verification of SC when submitting payword.*

Proof: The proof follows from the preimage resistance property of cryptographic hash functions. During the batch payment initialization, the buyer records the tip $H^{n_b}(s)$ on the blockchain. For anyone to successfully forge a (p, n_p)-pair, it must satisfy $H^{n_p}(p) = H^{n_b}(s)$ where $n_p \leq n_b$. If there existed a polynomial-time algorithm FindPNPair$(H^{n_b}(s))$ that could generate such a pair (p, n_p), it would directly violate the preimage resistance of the hash function. Therefore, forging a valid (p, n_p)-pair that passes SC's verification is at least as hard as finding a preimage for a cryptographically secure hash function, which is assumed to be computationally infeasible. □

Theorem 2: *With Lemma 1, the data owner can receive payment only after both successfully transferring the desired data to the buyer and being verified as LO.*

Proof: We prove this theorem by demonstrating that all potential malicious behaviors by the data owner to obtain payment without proper data delivery will fail.

Firstly, a malicious owner might attempt to suspend data transfer after receiving partial payment. However, this strategy fails as the buyer can immediately halt payword delivery upon detecting such behavior.

Secondly, a malicious owner might attempt to send incorrect data. This attack is prevented because the buyer has already received the hash values H_d of all data blocks at step ⑧. The buyer can verify both the integrity of received data by comparing its hash with H_d and validate that the data content meets his requirements $\mathcal{D}$.

Thirdly, an owner might attempt to forge a valid (p, n_p)-pair through brute force calculation to pass SC's verification. However, as established in Lemma 1, this is computationally infeasible given a cryptographically secure hash function.

Fourthly, an owner might attempt to misappropriate data from others and resell it as a new NFT, which represents the copyright protection problem addressed in this paper. However, as shown in Algorithm 4, SC verifies that only LO who has succeeded in the copyright challenge can submit paywords for payment, which prevents such misappropriation. □

Theorem 3: *MO cannot succeed in the copyright challenge when LO participates.*

Proof: MO may attempt to succeed in the copyright challenge through two primary strategies.

Firstly, they could attempt to manipulate the challenge process by sending the unauthorized challenge data D_c to the buyer while simultaneously submitting fraudulent challenge vectors V_c' to the blockchain. This strategy would theoretically prevent CC from finding any matching vector set V_s, as they would be attempting to match against forged vectors. However, this manipulation is prevented by the validity challenge step (Algorithm 2). During this step, the buyer independently recomputes the challenge vectors V_c from the received D_c and verifies them against those recorded on the blockchain. Any discrepancy between the computed vectors and the blockchain-recorded vectors V_c' will set $isDataOK$ to False, causing the copyright challenge to fail.

Secondly, MO might attempt to modify the original misappropriated data through minimal modifications. With n_f modified blocks in a dataset of size n_d, the probability of success is bounded by:

$$\Pr(\text{MO succeeds}) = 1 - \frac{\binom{n_d - n_f}{n_c}}{\binom{n_d}{n_c}},$$

where $\binom{n}{m}$ represents the binomial coefficient. Let $r_c = n_c/n_d$ denote the challenge ratio and $r_f = n_f/n_d$ the fake data ratio. As shown in Fig. 5, to achieve a meaningful probability of success, MO would need to modify a substantial portion of the data ($r_f \gtrsim 10\%$ for large datasets and $r_f \gtrsim 60\%$ for smaller datasets). This level of modification far exceeds the scope of minimal modifications considered in this paper. Therefore, our scheme effectively prevents copyright infringement through minimal modifications of the original data. □

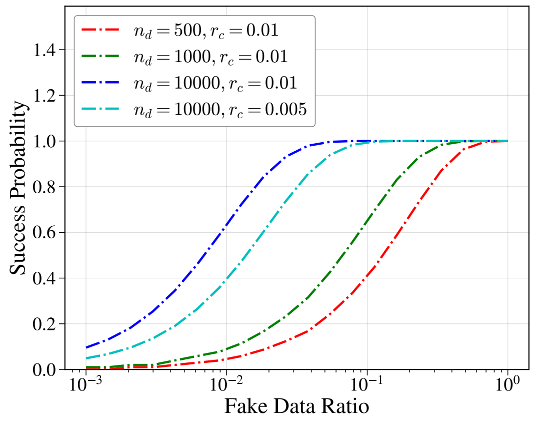


Fig. 5: Success probability of copyright challenge versus fake data ratio for MO.

Lemma 2: *Given a cryptographically secure hash function, it is computationally infeasible for anyone with polynomial-time capabilities to generate a valid Merkle proof for a modified vector.*

Proof: According to Algorithm 3, the verification of a vector's inclusion in a Merkle Tree is performed using VerifyMerkleProof$(H(v), p, r)$, where v is the vector, p is its Merkle proof, and r is the Merkle root. For a modified vector v', generating a valid Merkle proof p' would require satisfying VerifyMerkleProof$(H(v'), p', r) = $ True.

The verification function consists of iterative applications of MidHash$(H_l|H_r) = H(H_l|H_r)$, where H_l and H_r are the hash values of left and right child nodes in the Merkle Tree. If a polynomial-time algorithm existed to generate a valid proof p' for v', it would imply the existence of an algorithm FindChildren(MidHash$(H_l|H_r)) = (H_l', H_r')$ that satisfies:

$$\text{MidHash}(H_l|H_r) = \text{MidHash}(H_l'|H_r').$$

This would be equivalent to finding a second preimage for the hash function in polynomial time, which contradicts the security properties of a cryptographically secure hash function. Therefore, generating a valid Merkle proof for a modified vector is at least as hard as finding a second preimage collision in the underlying hash function. □

Theorem 4: *With Lemma 2, CC without the legitimte data cannot succeed in the copyright challenge against LO.*

Proof: Let us proceed by contradiction. Suppose CC could successfully pass the copyright challenge against LO. According to Algorithm 3, this would require CC to possess a vector set V_s satisfying two conditions: 1) The hash values of V_s must correspond to a set of Merkle roots R_s that were recorded on the blockchain earlier than LO's roots. 2) The vectors in V_s must be sufficiently similar to the challenge vectors V_c published by LO on the blockchain.

Since the challenge vectors V_c are randomly selected by the buyer and CC does not possess the legitimate data, the probability of CC naturally having such a matching vector set V_s is negligible. The only alternative would be for CC to: 1) find an existing vector set V_s' with earlier recorded Merkle roots R_s'. 2) modify V_s' to match V_c while maintaining valid Merkle proofs P_s for R_s'

However, by Lemma 2, generating valid Merkle proofs for modified vectors is computationally infeasible given a cryptographically secure hash function. Therefore, our initial assumption must be false, and CC cannot successfully complete the copyright challenge against LO. □

Theorem 5: *The data buyer cannot obtain ownership of the data NFT without paying both the full data price and the NFT ownership transfer fee.*

Proof: Let us consider two scenarios in the payment verification process as described in Algorithm 4: 1) For regular batch payments (when $N_p + n_p < n_d$), SC only executes the payment transfer operation. 2) For the final batch with NFT transfer (when $N_p + n_p == n_d$ and $dtype ==$ TDO), SC executes an atomic operation that combines the token transfer and NFT ownership transfer operations together. This atomic operation ensures that either both the payment and NFT ownership are transferred successfully, or the entire transaction is reverted if either transfer fails. Therefore, it is impossible for the buyer to obtain NFT ownership without completing both the full data payment and the NFT transfer fee payment. □

Combined with Theorems 1-5, this completes the proof of enhanced exchange fairness as defined in Section IV-C.

B. Data Privacy

Theorem 6: *With Theorems 1 and 2 as foundational, NMFT ensures the preservation of data privacy.*

Proof: Let us analyze data privacy across each phase of the protocol.

During the Setup, Mint data NFT and Request data phase, the complete dataset remains exclusively in the possession of the data owner, with no exposure to any other participants.

In the Challenge phase, data exposure is strictly limited. The buyer receives only a small random subset D_c for verification purposes, while other CCs access solely the feature vectors V_c, not the actual challenge data D_c.

For the Batch payment phase, based on Theorems 1 and 2, the buyer is entitled to receive data in proportion to the payment made except for a possible small batch of data obtained for free.

Throughout the entire protocol, complete data access is restricted to legitimate participants, data exposure is minimized and controlled, and unauthorized access is prevented through the payment mechanism. In a summary, the data privacy defined in Sec. IV-C is satisfied in the proposed NMFT protocol. □

VII. Experimental Results

A. Implementation

To evaluate the performance of our NMFT data trading protocol, we conducted a series of experiments on the Ethereum Sepolia testnet [42]. Sepolia is a proof-of-stake testnet that closely mimics the behavior of the Ethereum mainnet, providing a realistic environment for our tests. We utilized the Hardhat development framework (version 2.22.11) to deploy and interact with our smart contracts, which were written in Solidity (version 0.8.27). Our smart contracts are available at https://github.com/tenondvpn/nmft.

B. Metrics

To rigorously evaluate the efficiency and scalability of NMFT protocol, we defined and measured the following key metrics across various experimental conditions:

- **Transaction performance:** This includes both gas consumption (measured in gas units) and transaction latency (time from submission to confirmation) for each of the five phases, as described in Section IV. These metrics provide insights into the cost-efficiency and responsiveness of our protocol in different operations.
- **Scalability:** According to Algorithm 3, the challenge size n_c will affect the performance of the Challenge phase (P4), as both the data owner and CC need to upload n_c feature vectors to the smart contract for verification and similarity calculation. According to Algorithm 4, the new completed batch number n_p, or the batch number n_b will impact the performance of the Batch payment phase (P5). Threrefore, we evaluated how transaction performance scales with these two key protocol parameters.

C. Optimization

For feature extraction during the challenge phase, we use pre-trained models that output high-dimensional vectors (384-dim for MiniLM [35], 768-dim for MPNet [34] and DINOv2-base [36]). However, these floating-point vectors pose challenges in blockchain environments: Solidity lacks native floating-point support, and storing high-dimensional vectors incurs substantial gas costs.

We address these challenges using Locality-Sensitive Hashing (LSH) [43] to compress the high-dimensional vectors (including v_{A-D} in Fig. 3 and those denoted by lowercase v with various subscripts) into single uint256 integers. This reduces storage from $n_v \cdot n_c$ floating points to n_c integers per challenge, where n_v denotes the dimension of the vectors before LSH. For a vector v, the compression is:

$$\text{LSH}(v) = \sum_{i=1}^{n_v} \mathbb{1}(v \cdot r_i > 0) \cdot 2^i \bmod 2^{256},$$

where r_i are random but fixed projection vectors of the same dimension as v, $\mathbb{1}(\cdot)$ is the indicator function that outputs 1 if the condition is true and 0 otherwise, and the result $\text{LSH}(v)$ is a uint256 integer.

LSH preserves similarity relationships as similar vectors produce similar bit patterns. For compressed vectors v_1 and v_2 (each as a uint256 integer), the function CalcSimilarity in Algorithms 1 and 3 is implemented as:

$$\text{CalcSimilarity}(v_1, v_2) = \frac{L - \text{HD}(v_1 \oplus v_2)}{L} \times 100\%,$$

where $L = 256$ is the length of compressed vectors, $\text{HD}(\cdot)$ denotes the Hamming distance (number of set bits) of a binary number, and $\oplus$ represents the bitwise XOR operation. This computation is highly efficient and gas-friendly in smart contract environments, as it only involves basic bitwise operations.

We evaluated this scheme on both text and image data. For text, we used 200 samples across 5 similarity levels (ultra-high, high, medium-high, medium, and low similarity) with MiniLM and MPNet models. For images, we used random selected 100 CIFAR-10 images [44], each with 9 different modifications[1], resulting in 1000 total images. Features were extracted using DINOv2 [36] and Swin [37] models.

As shown in Fig. 6 and Fig. 7, LSH compression exhibits an asymmetric effect: it increases similarities below approximately 0.8 but slightly decreases those above. This benefits our copyright challenge mechanism. For example, with threshold $\eta = 0.9$, only genuinely similar data passes the challenge while false positives are prevented.

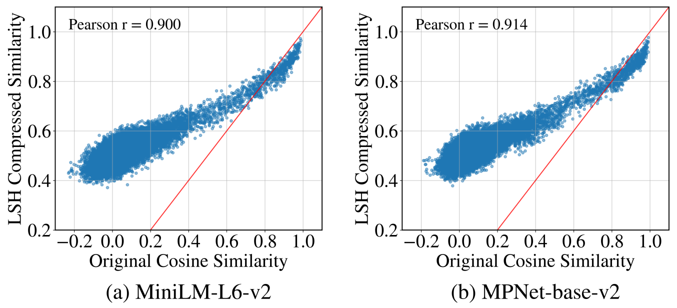


Fig. 6: LSH compressed similarity versus original cosine similarity for different text embedding models: (a) MiniLM-L6-v2 [35] and (b) MPNet-base-v2 [34].

To further reduce costs, we store compressed vectors as contract events rather than contract state, keeping only their hash values for verification.

D. Transaction performance

To evaluate the performance of our NMFT protocol, we performed experiments to measure gas consumption and transaction latency in different phases of the protocol. For our

[1]Including rotation, brightness/contrast adjustment, translation, color saturation, sharpness, cropping, scaling, and noise.

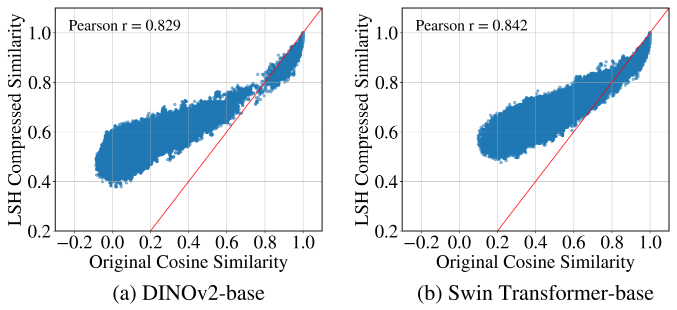


Fig. 7: LSH compressed similarity versus original cosine similarity for different image embedding models: (a) DINOv2-base [36] and (b) Swin Transformer-base [37].

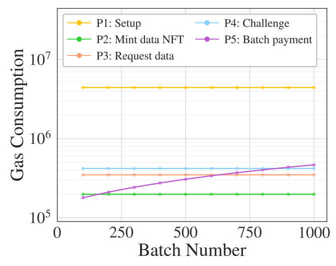
baseline experiments, we set the challenge size $n_c = 10$ and desired batch number $n_b = 100$. All smart contract functions within each phase are executed 10 times. As illustrated in Table II, the gas consumption across phases P1 to P5 demonstrates high consistency over the 10 test iterations, with a notable exception in P2 (Mint data NFT). Phases P1 and P3 exhibit invariant gas consumption, with identical minimum and maximum values. For P2, we observed that the first execution consumed significantly more gas (232,532) compared to subsequent executions (around 198,332). This is typical behavior in smart contracts, where the first execution of certain operations (like minting the first NFT) often requires additional gas for initialization processes. Phases P4 and P5 show minimal variation, with differences between minimum and maximum values of 590 and 12 gas units respectively. In contrast to the relatively stable gas consumption, the latency data exhibits considerable fluctuations across all phases. These variations in transaction latency are primarily attributed to the inherent network volatility of the Sepolia testnet.

TABLE II: GAS CONSUMPTION AND LATENCY BY PHASE.

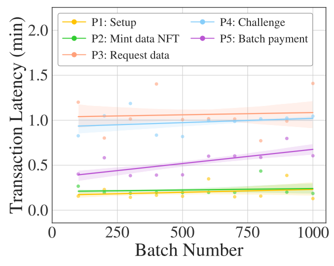
Phase	Gas consumption		Latency (min)	
	Min	Max	Min	Max
P1	4,421,929	4,421,929	0.129	0.386
P2	198,332	232,532	0.185	0.405
P3	349,072	349,072	0.793	1.461
P4	419,361	419,951	0.787	1.389
P5	179,230	179,242	0.382	0.834

Fig. 9 presents a breakdown of gas consumption and transaction latency across the five phases (P1-P5) of our NMFT protocol. This visualization is based on the median values of gas consumption and latency from 10 experimental runs for each phase.

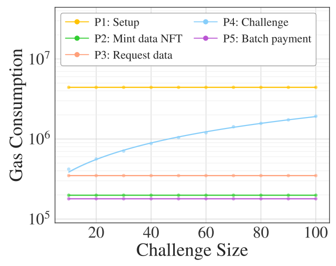
The breakdown of gas consumption reveals that P1 dominates, consuming 79% of the total gas. This is expected due to the complex initialization processes and state variable setups involved in deploying the smart contract. It is important to note that P1 only needs to be executed once during the initial deployment of the smart contract, and this high gas cost is not incurred in subsequent operations of the protocol. P4 follows at 8%, while P3 accounts for 6%. P2 consumes 4% of the gas, and P5 uses the least at 3%. These latter phases represent the ongoing operational costs of the protocol.



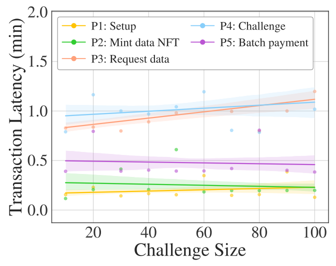
(a) Gas consumption varying batch number.



(b) Transaction latency varying batch number.



(c) Gas consumption varying challenge size.



(d) Transaction latency varying challenge size.

Fig. 8: Gas consumption and transaction latency under different batch numbers and challenge sizes.

In contrast, the latency breakdown shows a more balanced distribution. P3 takes the largest share at 37% of the total latency, followed by P4 at 35%. This suggests that these phases involve more time-consuming operations. P5 accounts for 15% of the latency, while P1, despite its high gas consumption, only contributes 6% to the overall latency. And P2 shows a latency of 7%.

This analysis highlights the disparity between gas consumption and time efficiency across different phases. Although P1 dominates in terms of gas consumption, it is relatively quick to execute. In contrast, P3 and P4 are more time-consuming but less gas-consuming.

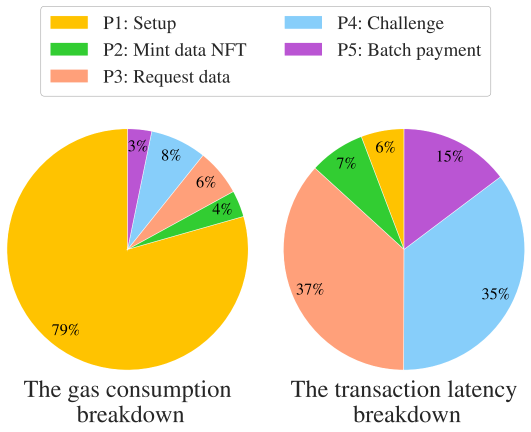


Fig. 9: Breakdown of gas consumption and transaction latency across different phases of the NMFT protocol, based on median values from 10 experimental runs.

E. Scalability

To evaluate NMFT's scalability, we conducted two analyses: (1) varying batch number n_b from 100 to 1000 in steps of 100, with fixed $n_c = 10$, and (2) varying challenge size n_c from 10 to 100 in increments of 10, with fixed $n_b = 100$.

As shown in Fig. 8a and 8c, gas consumption for P5 and P4 increases linearly with n_b and n_c respectively, while other phases remain constant. When n_b increases 10-fold (100 to 1000), P5's gas consumption grows only 2.6 times (179,230 to 464,646 gas units). Similarly, a 10-fold increase in n_c (10 to 100) leads to a 4.6-fold increase in P4's gas consumption (419,878 to 1,926,618 gas units). This sublinear growth suggests efficient handling of larger batches and challenges.

Transaction latency remains stable across different n_b and n_c values (Fig. 8b and 8d), as network latency dominates processing time.

The scaling behavior of n_b is advantageous for buyers to seek large data volumes. For a fixed total data volume requirement, buyers tend to purchase fewer batches by increasing the data volume per batch. However, it may be unfavorable for the owners, as larger volumes could lead to greater losses if buyers refuse to send a payword upon receiving the data.

On the other hand, the scaling behavior of n_c permits a wide range of challenge size. Larger n_c can benefit MOs who mix fake data, increasing their chances of winning copyright challenges, but this incurs higher gas fees for uploading larger V_c. In contrast, smaller n_c favor buyers and CC by enhancing their chances of obtaining stakes from MOs while avoiding higher gas fees.

VIII. Conclusion

In this paper, we present NMFT, a blockchain-based data trading protocol that addresses the critical issue of copyright protection in NFT-based data exchanges. The core of our protocol is the Merkle Feature Tree, which incorporates an AI-powered feature layer above the data layer of the traditional Merkle Tree. Copyright verification is achieved through a three-step process: verifying the Merkle proofs of features, comparing the on-chain timestamps of Merkle roots, and computing feature similarity. To make this verification practically feasible on blockchain, we employ Locality-Sensitive Hashing (LSH) to compress high-dimensional floating-point feature vectors into single uint256 integers, which are recorded as contract events in the blockchain logs to further optimize storage costs.

Extensive experiments with multiple well-established feature extraction models for both text and image data demonstrate that our LSH compression scheme effectively preserves similarity relationships. Particularly, when setting a high similarity threshold, the compression characteristics ensure that only genuinely similar data can pass the copyright challenge while preventing false positives. Experimental results on the Ethereum Sepolia testnet show that NMFT demonstrates good scalability with sublinear growth in gas consumption as batch numbers and challenge sizes increase, while maintaining stable transaction latency. These characteristics make NMFT particularly suitable for large-scale data trading scenarios where copyright protection is crucial.

References

[1] L. Faridoon, W. Liu, and C. Spence, "The impact of big data analytics on decision-making within the government sector," *Big Data*, vol. 0, no. 0, p. null, 2023, pMID: 38193755. [Online]. Available: https://doi.org/10.1089/big.2023.0019

[2] A. O. Adewusi, U. I. Okoli, E. Adaga, T. Olorunsogo, O. F. Asuzu, and D. O. Daraojimba, "Business intelligence in the era of big data: a review of analytical tools and competitive advantage," *Computer Science & IT Research Journal*, vol. 5, no. 2, pp. 415–431, Feb. 2024. [Online]. Available: https://www.fepbl.com/index.php/csitrj/article/view/791

[3] X. Wu, W. Li, and H. Tu, "Big data and artificial intelligence in cancer research," *Trends in Cancer*, vol. 10, pp. 147–160, 2023.

[4] Datacoup, "Datacoup," https://datacoup.com/, 2024, accessed: Aug. 26, 2024.

[5] Qlik, "Qlik," https://www.qlik.com/, 2024, accessed: Aug. 26, 2024.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[7] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[8] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," 2014, white paper, vol. 3, no. 37, pp. 2–1.

[9] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[10] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. ACM, 2019, pp. 347–356.

[11] Q. Wang, R. Li, Q. Wang, and S. Chen, "Non-fungible token (nft): Overview, evaluation, opportunities and challenges," 2021, arXiv:2105.07447.

[12] R. Song, S. Gao, Y. Song, and B. Xiao, ": A traceable and privacy-preserving data exchange scheme based on non-fungible token and zero-knowledge," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 2022, pp. 224–234.

[13] S. Ranjbar Alvar, M. Akbari, D. M. X. Yue, and Y. Zhang, "Nft-based data marketplace with digital watermarking," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 4756–4767. [Online]. Available: https://doi.org/10.1145/3580305.3599876

[14] O. P. Foundation, "Ocean protocol: Tools for the web3 data economy," https://oceanprotocol.com/tech-whitepaper.pdf, 2022, accessed: Aug. 26, 2024.

[15] Itheum.io, "Itheum whitepaper," https://dev.to/itheum/itheum-data-dex-whitepaper-ooo, 2021, accessed: Aug. 26, 2024.

[16] Datum, "Datum whitepaper," https://datum.org/assets/Datum-WhitePaper.pdf, 2017, accessed: Aug. 26, 2024.

[17] Cointelegraph, "Ai has a role to play in detecting fake nfts," 2023, accessed: 2024-08-27. [Online]. Available: https://cointelegraph.com/news/ai-has-a-role-to-play-in-detecting-fake-nfts

[18] Opensea, "Opensea," https://opensea.io/, 2024, accessed: Aug. 26, 2024.

[19] S. B. Journal, "Opensea self-reports that 80% of its nfts were unoriginal or illegitimate," 2022, accessed: Aug. 26, 2024. [Online]. Available: https://www.sportsbusinessjournal.com/Daily/Issues/2022/01/31/Technology/opensea-self-reports-that-80-of-its-nfts-were-unoriginal-or-illegtimate.aspx

[20] Opensea, "What is opensea's copymint policy?" 2024, accessed: Aug. 26, 2024. [Online]. Available: https://support.opensea.io/en/articles/8867065-what-is-opensea-s-copymint-policy

[21] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 967–984.

[22] S. Delgado-Segura, C. Pérez-Solà, G. Navarro-Arribas, and J. Herrera-Joancomartí, "A fair protocol for data trading based on bitcoin transactions," *Future Generation Computer Systems*, vol. 107, pp. 832–840, 2020.

[23] C. Chenli, W. Tang, H. Lee, and T. Jung, "Fair2trade: Digital trading platform ensuring exchange and distribution fairness," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–16, 2024.

[24] P. Abla, T. Li, D. He, H. Huang, S. Yu, and Y. Zhang, "Fair and privacy-preserved data trading protocol by exploiting blockchain," *IEEE Transactions on Information Forensics and Security*, pp. 1–1, 2024.

[25] D. Sheng, M. Xiao, A. Liu, X. Zou, B. An, and S. Zhang, "Cpchain: A copyright-preserving crowdsourcing data trading framework based on blockchain," in *2020 29th international conference on computer communications and networks (ICCCN)*. IEEE, 2020, pp. 1–9.

[26] Y. Xiang, W. Ren, T. Li, X. Zheng, T. Zhu, and K.-K. R. Choo, "A multi-type and decentralized data transaction scheme based on smart contracts and digital watermarks," *Journal of network and computer applications*, vol. 176, p. 102953, 2021.

[27] B. Wang, W. Huang, B. Li, Y. Yuan, F. Yang, and Z. Hu, "Blockchain-based medical image data trading platform with copyright and privacy protection," in *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2024, pp. 224–229.

[28] B. Wang, B. Li, Y. Yuan, C. Dai, Y. Wu, and W. Zheng, "Cpdt: A copyright-preserving data trading scheme based on smart contracts and perceptual hashing," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 968–975.

[29] X. Chen, A. Yang, J. Weng, Y. Tong, C. Huang, and T. Li, "A blockchain-based copyright protection scheme with proactive defense," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 2316–2329, 2023.

[30] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] R. C. Merkle, "Secrecy, authentication, and public key systems," Ph.D. Thesis, Stanford University, Stanford, CA, USA, 1979.

[34] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mpnet: Masked and permuted pre-training for language understanding," *Advances in neural information processing systems*, vol. 33, pp. 16 857–16 867, 2020.

[35] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5776–5788, 2020.

[36] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2023, arxiv:2304.07193.

[37] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[38] J. Benet, "Ipfs - content addressed, versioned, p2p file system," 2014, arXiv:1407.3561.

[39] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, p. 770–772, nov 1981. [Online]. Available: https://doi.org/10.1145/358790.358797

[40] R. L. Rivest and A. Shamir, "Payword and micromint: Two simple micropayment schemes," in *Proceedings of the 4th Security Protocols International Workshop*, ser. Lecture Notes in Computer Science, vol. 1189. Berlin: Springer-Verlag, 1996, pp. 69–87.

[41] M. Elsheikh, J. Clark, and A. M. Youssef, "Short paper: Deploying payword on ethereum," in *Financial Cryptography and Data Security*, A. Bracciali, J. Clark, F. Pintore, P. B. Rønne, and M. Sala, Eds. Cham: Springer International Publishing, 2020, pp. 82–90.

[42] Ethereum Sepolia Implementation Reference, "Sepolia testnet," https://github.com/eth-clients/sepolia, 2024, accessed: Aug. 26, 2024.

[43] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 604–613.

[44] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.