BIMW: Blockchain-Enabled Innocuous Model Watermarking for Secure Ownership Verification

Xinyun Liu and Ronghua Xu *

Posted Date: 22 September 2025

doi: 10.20944/preprints202509.1649.v1

Keywords: Ownership Verification; Model Watermarking; Deep Neural Networks; Blockchain; Edge Computing

Article

BIMW: Blockchain-Enabled Innocuous Model Watermarking for Secure Ownership Verification

Xinyun Liu and Ronghua Xu *

Department of Applied Computing, Michigan Technological University, Houghton, MI 49931, USA
* Correspondence: ronghuax@mtu.edu

Abstract

The integration of Artificial Intelligence (AI) and edge computing gives rise to Edge Intelligence (EI), which offers effective solutions to the limitations of traditional cloud-based AI; however, deploying models across distributed edge platforms raises concerns regarding authenticity, thereby necessitating robust mechanisms for ownership verification. Currently, backdoor-based model watermarking techniques represent a state-of-the-art approach for ownership verification; however, their reliance on model poisoning introduces potential security risks and unintended behaviors. To solve this challenge, we propose BIMW, a blockchain-enabled innocuous model watermarking framework that ensures secure and trustworthy AI model deployment and sharing in distributed edge computing environments. Unlike widely applied backdoor-based watermarking methods, BIMW adopts a novel innocuous model watermarking method called Interpretable Watermarking (IW), which embeds ownership information without compromising model integrity or functionality. In addition, BIMW integrates a blockchain security fabric to ensure the integrity and auditability of the watermarked data during storage and sharing. Extensive experiments are conducted on a Jetson Orin Nano board, which simulates edge computing environments. The results demonstrate our framework's superior performance in terms of effectiveness and innocuousness, as well as its robustness against potential model watermarking attacks.

Keywords: ownership verification; model watermarking; deep neural networks; blockchain; edge computing

1. Introduction

Recent years have witnessed rapid progress in the Internet of Things (IoT), edge computing, and cloud computing. Breakthroughs in artificial intelligence (AI), especially deep learning (DL), have further accelerated this trend [1,2]. Edge intelligence (EI), which integrates AI with edge computing, has shown great potential to overcome limitations of traditional cloud-based AI systems [3]. AI model deployment on edge devices enables autonomous and resilient systems for smart cities [4], while the integration of AI into edge platforms provides additional key benefits [5]. First, edge computing enables AI models to operate locally, facilitating real-time processing with minimal latency. Second, performing computations closer to the data source enhances data privacy by reducing the need to transmit sensitive information to centralized servers [6]. Moreover, decreasing reliance on cloud services lowers data transmission costs and mitigates network bandwidth limitations, thereby improving the scalability and efficiency of AI applications [7]. While we primarily consider edge intelligence (EI) as the general application scenario, the proposed framework is equally applicable in cloud-based environments.

Advancing AI to distributed edge platforms introduces significant security challenges due to the dynamic and heterogeneous nature of edge computing. Typically, cloud servers manage pretrained AI models and deploy them to remote edge environments. However, models on edge platforms are susceptible to model theft attacks, such as model extraction or weight stealing, which can replicate a

 doi:10.20944/preprints202509.1649.v1

model's functionality and lead to intellectual property (IP) infringement [8]. In addition, an adversary also launches poisoning attacks on original models, such as manipulating the training data or the model parameters [9]. As a result, these unauthorized AI models can be published on AI model marketplaces and then integrated into diverse smart applications. These issues not only compromise the model's commercial value and security but they also undermine the trustworthiness and reliability of AI ecosystems. Therefore, AI model authenticity and ownership verification are critical to protect model IP and prevent unauthorized access and even malicious use.

The fundamental concept of AI model ownership verification is to embed a specific pattern or structure into the model as a unique identifier (watermark), which can later be extracted to demonstrate ownership when necessary during the inference stage [10]. The verification process typically involves three key phases: watermark embedding, extraction, and validation [11]. In the embedding phase, watermark information is incorporated into the model either during training or through post-processing to establish ownership. Embedding techniques can operate at the parameter level, such as encoding the watermark within model weights, or at the functional level, such as introducing a trigger-based watermark that produces specific outputs for designated inputs [12]. Furthermore, the embedding process must ensure imperceptibility, meaning that the watermark does not degrade the model's performance on its primary task.

During the watermark extraction phase, predefined methods are used to retrieve the watermark from the target model during inference or auditing. If the extracted watermark matches the original one, ownership of the model can be verified [13]. The extraction method typically corresponds to the embedding technique; for example, if the watermark is encoded within model weights, specific parameter patterns must be analyzed to extract it, whereas trigger-based watermarks require feeding specially crafted inputs to activate the watermark response [14]. Finally, in the ownership verification phase, the extracted watermark is compared against the original to confirm the model's legitimacy. This verification process should have legal enforceability, allowing it to serve as valid evidence in intellectual property protection. Additionally, the watermark must be robust against various attacks, such as model compression, fine-tuning, pruning, and adversarial attacks, ensuring that it remains intact and detectable even after modifications. An ideal model watermarking technique should meet several core requirements. First, it must exhibit robustness, meaning that the watermark remains extractable even if the model undergoes modifications such as pruning or fine-tuning [15]. Second, the embedding should be imperceptible, ensuring that it does not degrade the model's normal task performance. Furthermore, the watermark should possess security, making it difficult for attackers to remove, forge, or alter the embedded information. Finally, the ownership verification process must be efficient, ensuring that watermark embedding and extraction incur minimal computational overhead, making it suitable for resource-constrained edge devices [16–18].

Currently, backdoor-based watermarking methods are among the most advanced techniques for ownership verification [14]. These methods rely on backdoor trigger mechanisms, embedding watermarks by injecting backdoors into models so that they produce predetermined responses when exposed to specific trigger samples. However, despite their effectiveness, backdoor-based watermarking fundamentally relies on poisoning the model, introducing additional security risks [19]. For instance, if an attacker discovers the trigger pattern, they could exploit the backdoor to launch adversarial attacks against the model. Additionally, even if the trigger pattern has a minimal impact, it may still cause unexpected erroneous predictions, affecting the model's reliability. Moreover, as research on backdoor detection advances, backdoor-based watermarking techniques may become increasingly susceptible to detection and removal, compromising their long-term effectiveness.

Our Work. To solve these challenges, we propose BIMW, a novel blockchain-enabled innocuous model watermarking framework for secure ownership verification of AI models within distributed AI application ecosystems. As Figure 1 shows, owners rely on cloud servers to manage pretrained models and deploy models for authorized users. However, an adversary can publish unauthorized models (copied or manipulated) in AI applications. To ensure verifiable and robust model ownership verifica-

 doi:10.20944/preprints202509.1649.v1

tion, BIMW introduces an interpretable watermarking (IW) method that avoids model poisoning while maintaining robustness. Instead of embedding watermarks directly into model outputs, the cloud server (model owner) employs feature impact analysis algorithms to generate interpretation-based watermarks, along with corresponding trigger samples. Ownership is verified by using these trigger samples for the model inference and then comparing the resulting interpretations with the original watermark. Thus, users can confirm that models are published by trusted sources with authorized permissions.

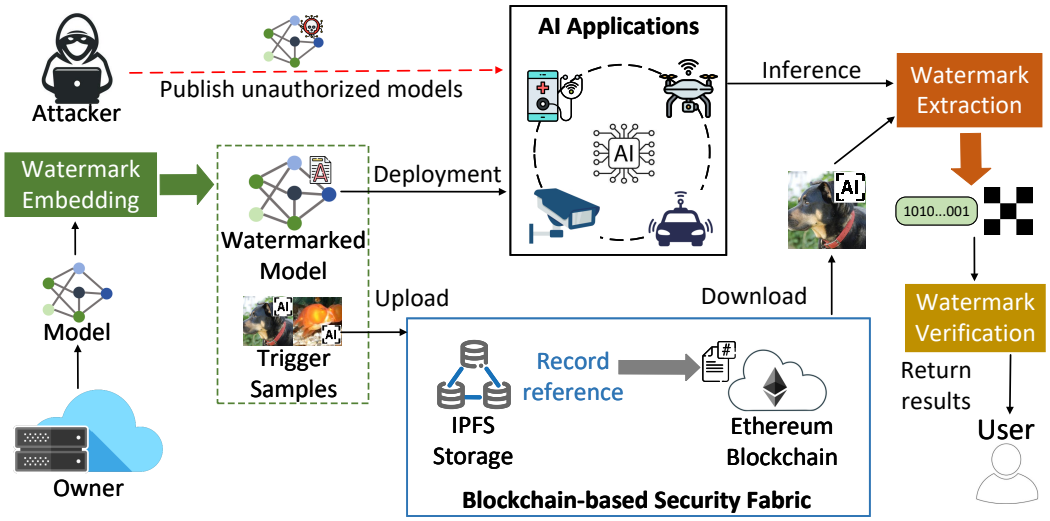

Figure 1. The system overview of BIMW framework.

To further enhance security and transparency of model ownership verification under a distributed network environment, BIMW integrates a blockchain-based security fabric that ensures the integrity and auditability of watermark data, models, and trigger samples during storage and distribution. As the bottom of Figure 1 shows, model owner saves models and trigger samples into a IPFS-based distributed storage and then records their reference information on a Ethereum Blockchain. Upon model deployment and inference stages, users (or owners) use reference information to identify any tampered models or trigger samples. Therefore, BIMW offers a decentralized, secure, and reliable alternative to traditional backdoor-based watermarking methods. Our implementation is publicly available at https://github.com/Xinyun999/BIMW.

The primary contributions are outlined as follows:

1) We propose the system architecture of BIMW, a blockchain-enabled innocuous model watermarking framework that ensures secure and trustworthy AI model deployment and sharing in distributed edge computing environments.
2) We demonstrate a Innocuous Model Watermarking method, which consists of IW embedding and feature impact analysis for watermark extraction.
3) We strengthen the data transmission security and transparency of watermark data, models, and trigger samples by using blockchain encryption technology.
4) We conduct comprehensive experiments by applying BIMW to various AI models. The results demonstrate its effectiveness and resistance against both watermark-removal and adaptive attacks and efficiency of model data authentication and ownership verification at edge computing platforms.

The remainder of this paper is structured as follows. Section 2 provides a brief overview of existing solutions for deep learning model watermarking and ownership verification. We also briefly describe interpretable machine learning techniques and Blockchain technology. Section 3 presents system framework of innocuous model watermarking method, emphasizing the synergic integration of IW and feature impact analysis (FIA) for watermark embedding, extraction, and ownership verification.

 doi:10.20944/preprints202509.1649.v1

Section 4 describes the prototype implementation and shows the experimental results to verify the effectiveness and performance of applying BIMW on edge computing platforms. Finally, Section 5 concludes this paper with a summary and some discussions about ongoing efforts.

2. Background Knowledge and Related Work

2.1. Edge Intelligence

With AI advancing rapidly, traditional cloud computing faces growing challenges like high latency, bandwidth consumption, and data privacy risks [5]. Edge Intelligence, which integrates IoT, edge computing, and AI technology, offers a promising solution. By directly deploying AI models on IoT devices that are close to the network of the edge, edge intelligence demonstrates several merits, such as local data processing, improving real-time responsiveness, enhancing privacy, reducing bandwidth use, and increasing system stability. This technology is widely used in intelligent surveillance, autonomous driving, smart healthcare, industrial inspection, and IoT systems, allowing efficient operation in resource-limited environments [1,20,21]. However, deploying AI models on edge devices also raises model theft risks. Limited computational power and weak security make these devices vulnerable to model extraction and reverse engineering, leading to intellectual property theft and unauthorized use. As a result, ownership verification has become a key challenge in ubiquitous smart applications based on edge intelligence.

2.2. Model Watermarking and Ownership Verification

Model watermarking has emerged as a promising technique for embedding identifiable information into neural networks, enabling model ownership verification while preserving model functionality [22]. The watermark-based ownership verification typically includes three phases: embedding, extraction, and verification [11]. Watermarks can be embedded during training or post-processing, either at the parameter level (e.g., within model weights) or functional level (e.g., trigger-based watermarks) [12]. Extraction techniques align with embedding methods; for instance, weight-based schemes analyze parameter patterns, while trigger-based approaches use crafted inputs to elicit responses [14].

Model watermarking techniques can be broadly categorized into white-box, black-box approaches. White-box watermarking embeds a watermark directly into the model's parameters, such as weights and biases, making it retrievable only when internal access to the model is available [23]. This method provides strong security but requires direct access to the model's architecture and parameters. In contrast, black-box watermarking enables verification through query-based interactions with the model's API, embedding specific responses or backdoor triggers that can be used to verify ownership without requiring internal access [24].

The watermarking process generally consists of two key phases: embedding and extraction. During the embedding phase, an unique watermark, such as a specific set of activations, adversarial triggers, or modified training data, is introduced into the model during training. In the extraction phase, the embedded watermark is later retrieved to verify model ownership, either by examining internal parameters in a white-box setting or by evaluating specific outputs based on carefully crafted queries in a black-box setting. An effective model watermarking scheme must satisfy several key properties. Robustness ensures that the watermark remains intact even if the model undergoes transformations such as pruning, fine-tuning, or compression [11]. Fidelity guarantees that the watermark does not degrade the model's primary performance on its intended tasks. Security is crucial, as the watermark should be resistant to unauthorized removal or detection by adversaries. Finally, verifiability ensures that the ownership verification process is reliable and legally admissible if required.

Ownership verification establishes the legitimacy of a deep learning model by extracting and validating embedded watermarks through techniques such as signature verification, challenge–response mechanisms in black-box settings, and cryptographic hashing to ensure integrity and prevent tampering. Beyond technical validation, watermarks can also serve as admissible evidence in legal disputes, reinforcing accountability in cases of intellectual property theft [25]. Effective verification requires a

 doi:10.20944/preprints202509.1649.v1

balance of robustness, security, and efficiency, ensuring ownership can be proven without impairing model performance. In practice, model watermarking is widely applied to safeguard proprietary AI assets, detect unauthorized or unlicensed usage [13], and protect distributed models in federated learning and secure AI deployment. As deep learning continues to expand into critical domains, watermarking remains essential for protecting and managing AI ownership.

2.3. Interpretable Machine Learning

Interpretable machine learning (IML) aims to enhance the transparency and understanding of complex machine learning models, enabling users to comprehend how decisions are made [26]. As machine learning models, particularly deep neural networks, become increasingly intricate, the need for interpretability has grown in importance for ensuring trust, accountability, and fairness in AI-driven decision-making. Interpretability in machine learning can be categorized into intrinsic and post-hoc approaches. Intrinsic interpretability refers to models that are inherently transparent, such as decision trees, linear regression, and rule-based classifiers, where the reasoning behind predictions is easily understood. In contrast, post-hoc interpretability applies to complex models, such as deep neural networks, where explanations are generated after training using techniques like feature importance analysis, attention mechanisms, and surrogate models [27]. A key aspect of interpretable machine learning is explainability, which provides insights into how models arrive at specific predictions. Techniques such as SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model-agnostic Explanations) help quantify the contribution of input features to the model's output, making it easier to interpret predictions [28]. Additionally, saliency maps and attention mechanisms provide visualization-based explanations, particularly in deep learning applications such as image classification and natural language processing.

Despite its advantages, achieving interpretability often presents trade-offs. Highly interpretable models, such as linear regression and decision trees, may lack the predictive power of more complex models like deep neural networks. Conversely, while deep learning models excel in performance, their lack of transparency poses challenges in critical applications. Ongoing research in interpretable AI focuses on developing techniques that balance interpretability with accuracy, ensuring that models remain both effective and understandable [29]. Future directions in interpretable machine learning involve the integration of interpretability with adversarial robustness, fairness, and causality. Developing standardized evaluation metrics for interpretability, enhancing human-AI collaboration through interactive explanations, and incorporating domain-specific knowledge into interpretable models are key areas of interest.

2.4. Blockchain for Data Security

Edge computing poses significant challenges in ensuring data security, integrity, and transparency due to its decentralized nature. Traditional centralized storage systems are inherently vulnerable to single points of failure, data tampering, and unauthorized access. Blockchain technology, with its decentralized and immutable nature, provides a promising solution for securing AI model ownership and protecting sensitive data [25]. By leveraging blockchain as a decentralized authority, AI models can be registered, verified, and tracked in a tamper-proof manner, thereby enhancing the reliability of ownership verification mechanisms. One key advantage of blockchain technology is its immutability, ensuring recorded data cannot be altered or deleted without network consensus [1]. This is particularly critical for AI model ownership verification, as it allows watermarked models and ownership claims to be permanently and securely recorded on the blockchain. Any unauthorized modifications to the model or its metadata can be detected, mitigating intellectual property disputes. Blockchain's decentralized architecture enhances security by eliminating reliance on a central authority, reducing risks of data breaches and system failures through distributed data storage.

In addition to security and integrity, blockchain also offers auditability and transparency. Every transaction, including model registration, verification, and distribution, is permanently and verifiably stored [1], providing an indisputable record for legal or forensic purposes. Additionally, stakeholders can

 doi:10.20944/preprints202509.1649.v1

monitor model usage without compromising data confidentiality. Smart contracts further reinforce the system by automating access control, ensuring that only authorized parties can verify model ownership. For AI model watermarking, blockchain serves as a trusted repository for watermark registration and verification. The embedding details, verification protocols, and cryptographic hashes of the watermarked model can be securely stored on the blockchain, ensuring that ownership claims are tamper-proof. When disputes arise, the blockchain ledger acts as an authoritative source for validating ownership, reducing reliance on third-party verification services. This integration enhances the robustness of watermarking methods, making them more resistant to forgery and unauthorized modifications.

3. Methodology

In this section, we provide details of BIMW framework, especially for two sub-frameworks: Interpretable Watermarking (IW) model and Blockchain-based data verification. Building upon [15], our BIMW further enhances this line of work by improving both robustness and trustworthiness. In contrast to the previous scheme, our approach not only achieves better concealment and ownership verification but also strengthens the data transmission security and transparency of watermark data, models, and trigger samples through blockchain-based encryption technology.

3.1. Insight of Interpretable Watermarking

As discussed in Section 1, traditional backdoor-based watermarking techniques rely on model poisoning, which can introduce security risks and unintended behaviors. To address these challenges, we pose a critical question: Is there an alternative space where we can embed an inconspicuous watermark without affecting model predictions? Drawing inspiration from Interpretable Machine Learning (IML) and feature impact analysis (FIA), we propose Interpretable Watermarking (IW). Our key idea is that, rather than embedding watermarks directly within the model's predicted classes, we can utilize the interpretations generated by FIA algorithms as a hidden medium for watermarking.

Figure 2 presents an overview of the interpretable watermarking framework and contrasts it with conventional backdoor-based methods. Unlike approaches that modify a model's predicted class when a marked sample is introduced, IW employs feature impact analysis techniques to derive interpretations for these samples, embedding the watermark within these interpretability outputs. The IW process consists of three distinct phases: (1) embedding the watermark, (2) extracting the watermark, and (3) verifying ownership. Furthermore, we integrate blockchain technology to ensure the immutability and auditability of watermarked model records.

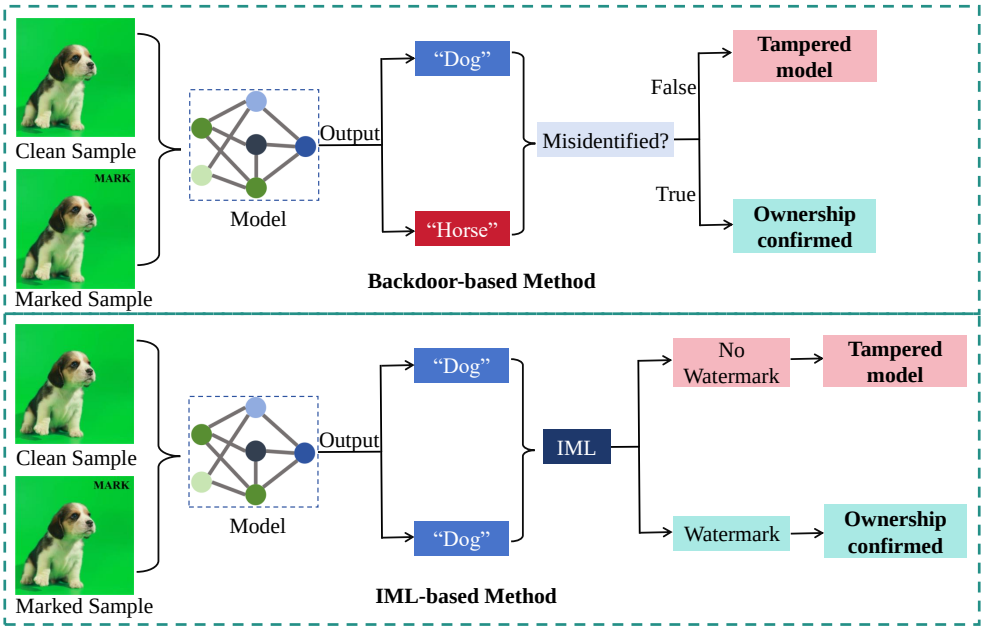

Figure 2. Interpretable model watermarking vs. backdoor-based model watermarking.

 doi:10.20944/preprints202509.1649.v1

3.2. Embedding the Watermark

As discussed in Section 1, an ownership verification mechanism aims to achieve three key objectives: effectiveness, robustness, and innocuous. During the watermark embedding phase, the watermark information is embedded into the model by adjusting the parameters ϕ of the trained model during training to establish ownership. Moreover, the model's original performance must be maintained after embedding the watermark [15]. Thus, the watermark embedding process can be framed as a multi-objective optimization problem based on these criteria, which is formally expressed as follows:

$$\min_{\phi} \mathcal{L}_1\big(g(I_{in} \cup I_{in}^T, \phi), c \cup c^T\big) + \alpha_1 \cdot \mathcal{L}_2\big(Interpret(I_{in}^T, c^T, \phi), W_M\big), \tag{1}$$

where ϕ denotes the parameters of the model and W_M represents the target watermark. The data I_{in} and labels c correspond to the clean dataset, whereas I_{in}^T and c^T represent the data and labels of the watermarked set. In our proposed method, the true labels of I_{in}^T are taken as c^T, while backdoor-based approaches rely on targeted, but incorrect, labels. The function Interpret(.) refers to an interpretability-based feature impact analysis technique employed in our watermarking method for watermark extraction, which will be detailed in Section 3-C. α_1 denotes a coefficient. Equation (1) consists of two components. The first term, $\mathcal{L}_1$, represents the model's loss function on the primary task, ensuring that the predictions on both the clean and marked datasets remain consistent, thus preserving the model's performance. The second term, $\mathcal{L}_2$, measures the discrepancy between the output interpretation and the target watermark. By optimizing $\mathcal{L}_2$, the model can align the interpretation more closely with the watermark. We use a hinge-like loss function for $\mathcal{L}_2$ because it has been shown to enhance the watermark's resilience against removal attacks. The hinge-like loss function is expressed as follows:

$$\mathcal{L}_2(\Theta, W_M) = \sum_{i=1}^{n} max(0, \theta - \Theta_i \cdot W_{M,i}), \tag{2}$$

where $\Theta = Interpret(I_{in}^T, c^T, \phi)$. The symbols Θ_i and $W_{M,i}$ represent the $i-th$ elemenets of Θ and W_M respectively. The parameter θ serves as a control factor, promoting the absolute values of the elements in Θ to exceed θ. By optimizing Equation (2), the watermark is embedded into the sign of the interpretation Θ.

3.3. Extracting the Watermark via Feature Impact Analysis

The goal of embedding a model watermark is to determine the optimal set of model parameters, denoted as $\hat{\phi}$, that minimizes Equation (2). To leverage the widely used gradient descent algorithm for this optimization, it is essential to develop a differentiable and model-agnostic method for feature impact analysis. Drawing inspiration from the well-known Local Interpretable Model-Agnostic Explanations (LIME) algorithm, we propose a LIME-based watermark extraction technique that generates feature impact interpretations for the trigger sample. LIME operates by generating local samples around a given input data point and assessing the significance of each feature based on the model's outputs for these samples. We adopt this fundamental idea while introducing modifications to tailor the algorithm for watermark embedding and extraction. The overall workflow of our watermark extraction approach is illustrated in Figure 3. Generally, the LIME-based watermark extraction process consists of three key phases: (1) regional sampling, (2) model prediction and assessment, and (3) obtaining the interpretation.

Phase 1: regional Sampling. Given that the input $I_{in} \in \mathbb{R}$, regional sampling aims to generate multiple samples that are locally adjacent to the trigger sample I_{in}^T. To begin, the input space is divided into n fundamental segments based on the length of the watermark $W_M \in \{-1, 1\}^n$. Neighboring features are grouped into a single segment, with each segment containing M/n features. Irrelevant features are excluded, as the goal is to extract a watermark rather than interpret all the features.

 doi:10.20944/preprints202509.1649.v1

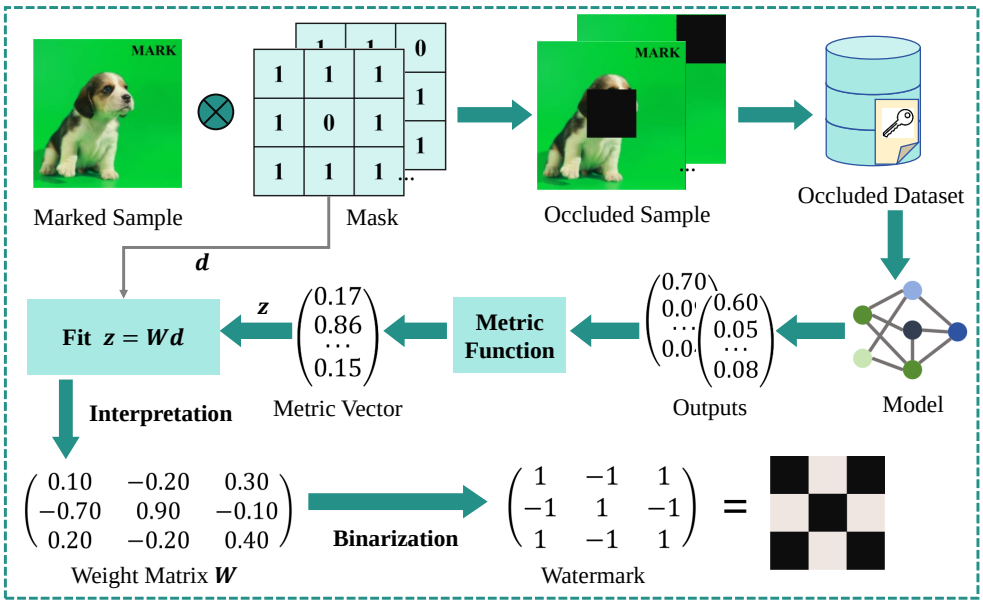

Figure 3. The core process of the watermark extraction algorithm based on feature impact analysis.

The core idea behind our algorithm is to determine which features have the greatest impact on the prediction of a data point by systematically masking these fundamental components. To achieve this, we first randomly generate s masks, denoted as M. Each mask in M is a binary vector (or matrix) of the same size as W_M. We refer to the $i-th$ mask in M as M_i, where for each i, $M_i \in \{-1,1\}^n$. Each element within a mask corresponds to a specific component of the input.

Next, we generate the masked samples I_m by applying random masks to the key components of the trigger sample, thereby creating a dataset. This masking process is represented by $\otimes$, *i.e.*, $I_m = M \otimes I_{in}^T$. Specifically, if the corresponding mask element M_i is 1, the related component in the input retains its original value. If M_i is 0, the component is replaced with a predefined value. Examples of the masked samples can be found in Figure 3.

Phase 2: Model Prediction and Assessment. During this phase, the masked dataset generated in Phase 1 is fed into the model, and the predictions $y = f(I_{in}, \phi)$ for the masked samples are obtained. In label-only settings, the predictions y are binarized to either 0 or 1, depending on whether the sample is classified correctly. Subsequently, a metric function $Dist()$ is used to assess the accuracy of the predictions in comparison to the ground-truth labels c^T, and the metric vector $v \in \mathbb{R}$ for the s masked samples is computed according to Equation (3).

$$v = Dist(y, c^T) \tag{3}$$

The metric function, denoted as $Dist()$, must be differentiable and capable of offering a quantitative assessment of the output. It can be tailored to suit specific deep learning tasks and prediction types. Given that most deep learning tasks typically have a differentiable metric function (such as a loss function), IW can be readily adapted for use in a wide range of deep learning applications.

Phase 3: Obtaining the Interpretation. Once the metric vector v is computed, the final phase of the watermark extraction process involves fitting a linear model to assess the significance of each component and determine their corresponding importance scores. We treat the metric vector v as z and the masks M as d. In practice, we apply ridge regression to enhance the robustness of the resulting weight matrix across various local samples. The weight matrix W, obtained through ridge regression, reflects the importance of each component. This weight matrix W for the linear model can be derived using the normal equation, as shown in Equation (4).

$$W = (M^T M + \tau E_s)^{-1} M^T v \tag{4}$$

 doi:10.20944/preprints202509.1649.v1

where τ denotes a hyper-parameter. E_s represents an $s \times s$ identity matrix. The watermark is embedded into the sign of the elements within the weight matrix. During the embedding process, we define the weight matrix W as $\Theta = Interpret(I_{in}^T, c^T, \phi)$ to optimize the watermark embedding loss function in Equation (1). As stated in Equation (4), the derivative of W with respect to v exists, and since the derivative of v with respect to the model parameters ϕ is also well-defined in DNN, the entire watermark extraction algorithm remains differentiable based on the chain rule. Thus, the watermark can be embedded into the model by leveraging the gradient descent algorithm to optimize Equation (1).

To further acquire the extracted watermark $\tilde{W}_M \in \{-1, 1\}^n$, we binarize the weight matrix W by applying the following binarization function bin(·).

$$\tilde{W}_{M,i} = \text{bin}(W_i) = \begin{cases} 1, & W_i \geq 0 \\ -1, & W_i < 0 \end{cases} \tag{5}$$

where $\tilde{W}_{M,i}$ and W_i are the i-th element of $\tilde{W}_M$ and W.

The key to applying watermark extraction across various tasks lies in designing the masking operation rule, denoted by $\otimes$, and the metric function $Dist()$. The masking operation is responsible for generating the masked samples, while the metric function evaluates the quality of predictions. To illustrate, we present an example implementation with image classification models, as shown in Figure 3. Specifically, the masking operation sets the pixels in the masked region to 0, while retaining the original values for the remaining pixels. Moreover, the metric function can be defined as the output that provides the predicted probability for the ground-truth class label.

3.4. Verifying Ownership

If the model owner identifies a suspicious model deployed by an unauthorized entity, they can determine whether it is a copy of the watermarked model by extracting the watermark from the suspicious model. The extracted watermark is then compared to the original watermark held by the model owner. This procedure is known as the ownership verification process for DNN models. For a suspicious model ϕ_{sus}, the model owner will begin by extracting the watermark $\tilde{W}_M$ using trigger samples and the watermark extraction algorithm based on feature impact analysis, as detailed in Section 3-C. The task of comparing $\tilde{W}_M$ with W_M is framed as a hypothesis testing problem, outlined as follows.

Theorem 1. Let $\tilde{W}_M$ represent the watermark extracted from the suspicious model, and W_M denote the original watermark. We define the null hypothesis Q_0 as: $\tilde{W}_M$ is independent of W_M, and the alternative hypothesis Q_1 as: $\tilde{W}_M$ is related or associated with W_M. The suspicious model can only be considered an unauthorized copy if Q_0 is rejected.

To verify ownership, we apply Pearson's chi-squared test and determine the corresponding p-value. If this p-value falls below a predefined significance threshold β, the null hypothesis is dismissed, confirming the model as the rightful intellectual property of its original owner. The ownership verification procedure is outlined in pseudocode form in Algorithm 1.

Algorithm 1 Ownership verification via hypothesis testing.

Require: Trigger set (I_{in}^T, c^T), suspicious model ϕ_{sus}, reference watermark W_M, significance level β.
Ensure: Boolean flag indicating whether ownership is verified.

```
W̃_M ← Interpret(I_in^T, c^T, φ_sus)
W̃_M ← bin(W̃_M)
p-value ← χ²-Test(W̃_M, W_M)
if p-value ≤ β then
    return True
else
    return False
end if
```

 doi:10.20944/preprints202509.1649.v1

3.5. Blockchain-based Data Verification

Both owners and users can rely on a Blockchain fabric to verify integrity of model data during storage and sharing. The Blockchain fabric uses IPFS [30] as distributed storage for model data distribution. The owner i can publish a model data which includes a watermarked model M_{IW} and associated trigger sample images $S_i = \{S_1, ..., S_k\}$ where k is the count of trigger sample images. After successfully uploading a model data onto IPFS network, the data owner can receive a set of unique hash-based Content Identifiers (CIDs), which can retrieve model data from IPFS network. The reference of a model data is represented as a Merkle tree of CIDs of watermarked model and trigger sample, which denotes as $MT\_root = MerkleTree(D_{S_1}, D_{S_2}, ...S_k, D_M)$. where D represents a CID. Finally, a sequential list of CIDs $D = (D_{S_1}, D_{S_2}, ...S_k, D_M)$ along with its merkle root MT_root will be saved on Blockchain through smart contracts.

At model retrieval stage, model users call smart contract's function to query D and MT_root from Blockchain. To verify the integrity of model data, the user simply reconstruct a Merkle tree of D and calculate its root hash MT_root'. Any Modification on the sequential order of D or content of watermarked model M_{IW} and trigger sample images S_i will lead to a different root hash value MT_root of the Merkle tree. Thus, integrity of received model data can efficiently verified by comparing MT_root' with proof information MT_root recorded on the Blockchain.

4. Experimental Results

In this section, we implement IW in a widely used deep learning task: image classification. We assess its effectiveness, safety, and uniqueness based on the objectives defined in Section 1. In addition, we also evaluate latency incurred by different operation stages in the model data sharing and authentication process. Furthermore, we examine IW's robustness against different watermark removal attacks.

Experimental Setup. The model is trained on using PyTorch and executed on four NVIDIA Tesla V100 GPUs. After obtaining the pre-trained model, we deployed it on a Jetson Orin Nano Super Developer Kit [31]. The Jetson Orin Nano is widely recognized as an effective edge device across various research domains, particularly in artificial intelligence (AI), robotics, computer vision, and embedded systems. Its combination of high computational power and energy efficiency makes it a suitable choice for real-time processing at the edge. We also implemented a prototype of Blockchain-based security fabric. We use Solidity [32] to develop smart contracts that record reference information of model and it samples and verify data integrity during sharing. Ganache [33] is used to set up a development Ethereum blockchain network. Truffle [34] is used to compile smart contracts and then deploy binary code on the development Blockchain network. We setup a private IPFS [35] network to simulate a distributed storage.

Watermark Definition. In the hypothesis test, we define the significance level as $\beta = 0.01$, meaning that if the p-value falls below 0.01, the null hypothesis is rejected. Besides, we compute the watermark success rate (WSR) to assess the similarity between the extracted and original watermarks. The WSR represents the proportion of bits in the extracted watermark that correctly match those in the original. It is defined by the following equation.

$$WSR = \frac{1}{n} \sum_{i=1}^{n} \chi\{\tilde{W}_{M,i} = W_{M,i}\} \tag{6}$$

where n represents the length of the watermark, and χ denotes the indicator function. A lower p-value and a higher WSR indicate that the extracted watermark $\tilde{W}_{M,i}$ closely resembles the original watermark $W_{M,i}$, signifying a more effective watermark embedding process.

4.1. Performance Evaluation on Image Classification Models

In this section, we perform experiments on the CIFAR-10 and a subset of the ImageNet datasets using the widely used convolutional neural network (CNN), ResNet-18. CIFAR-10 is a 10-class image

 doi:10.20944/preprints202509.1649.v1

classification dataset consisting of 32×32 color images. For the ImageNet dataset, we randomly select a subset of 80 classes, with 500 training images and 100 testing images per class. The images in ImageNet are resized to 224×224. Initially, we pre-train the ResNet-18 models on both the CIFAR-10 and ImageNet datasets for 400 epochs. Afterward, we fine-tune the models for 40 epochs to embed the watermark using IW. In line with the original LIME paper, we use the predicted probability of each sample's target class to form the metric vector v.

To assess the performance of IW, we implement various trigger set construction techniques inspired by different backdoor watermarking methods. These include: (1) Noise, where Gaussian noise is used as the trigger sample; (2) Patch, where a meaningful patch (e.g., 'MARK') is inserted into the images; and (3) Black-edge, which involves adding a black border around the images.

To assess both Effectiveness and Innocuity, The Table 1 and Table 2 reports the model's prediction accuracy (Pred Acc), the p-value from a hypothesis test assessing the statistical significance of accuracy differences, and the watermark success rate (WSR) under three different types of watermark triggers: Noise, Patch, and Black-edge. The watermark length varies from 32 to 256.

Table 1. Prediction accuracy, p-value, and WSR for model watermarking performance testing on the CIFAR-10 dataset.

L	*Metric*↓ *Trigger*→	*No WM*	*Noise*	*Patch*	*Black-edge*
32	Pred Acc.	91.36	91.28	91.25	91.23
	p-value	/	1.4×10^{-3}	1.3×10^{-3}	1.6×10^{-3}
	WSR	/	1.000	1.000	1.000
48	Pred Acc.	91.36	91.31	91.14	91.12
	p-value	/	7.6×10^{-4}	8.2×10^{-4}	6.3×10^{-4}
	WSR	/	1.000	1.000	1.000
64	Pred Acc.	91.36	91.22	91.29	91.26
	p-value	/	2.3×10^{-4}	3.1×10^{-4}	1.8×10^{-4}
	WSR	/	1.000	1.000	1.000
128	Pred Acc.	91.36	91.34	91.33	91.18
	p-value	/	9.8×10^{-5}	8.9×10^{-5}	6.4×10^{-5}
	WSR	/	1.000	1.000	1.000
256	Pred Acc.	91.36	91.26	91.18	91.31
	p-value	/	3.5×10^{-5}	4.7×10^{-5}	5.2×10^{-5}
	WSR	/	1.000	1.000	1.000

'L' represents the size of the embedded watermark.

Table 2. Model watermarking performance testing on a subset of the ImageNet dataset.

L	*Metric*↓ *Trigger*→	*No WM*	*Noise*	*Patch*	*Black-edge*
32	Pred Acc.	75.81	74.93	75.06	74.72
	p-value	/	1.2×10^{-3}	1.1×10^{-3}	1.4×10^{-3}
	WSR	/	1.000	1.000	1.000
64	Pred Acc.	75.81	74.98	75.15	74.89
	p-value	/	2.1×10^{-4}	3.6×10^{-4}	4.5×10^{-4}
	WSR	/	1.000	1.000	1.000
256	Pred Acc.	75.81	75.12	75.37	75.03
	p-value	/	1.4×10^{-5}	1.9×10^{-5}	2.3×10^{-5}
	WSR	/	1.000	0.999	0.999

'L' represents the size of the embedded watermark.

Across all experiments in Table 1 and Table 2, the baseline accuracy of the unwatermarked model (No WM) remains consistently at 91.36% for the CIFAR-10 dataset and 75.81% for the ImageNet dataset. When watermarks are embedded, the prediction accuracy shows only a negligible decline, indicating minimal impact on classification performance. The largest deviation is observed with the Black-edge trigger of length 48 in Table 1, where the accuracy slightly drops to 91.12%.

 doi:10.20944/preprints202509.1649.v1

The p-values are consistently low, ranging from 3.5×10^{-5} to 1.6×10^{-3} in Table 1, which are significantly lower than the threshold β. Furthermore, the WSR values are remain close to 1.000, indicating a nearly 100% success rate in watermark embedding and extraction across all configurations. The chi-squared test statistic is approximately proportional to $\frac{1}{n}$, where n represents the watermark length. Consequently, as n increases, the p-value correspondingly decreases. These findings clearly validate the effectiveness of IW in embedding watermarks into models while preserving classification accuracy.

To analyze the differences from Backdoor Watermarks, we conduct a comparative experiment, as shown in Table 3 and Table 4. To assess the level of harmlessness, we introduce the harmless degree H as an evaluation metric. Specifically, H is determined based on the accuracy obtained from both the benign testing dataset (x, y) and the trigger set (x_t, y_t) using their respective ground-truth labels, as defined below:

Table 3. Performance on CIFAR-10 dataset: watermark success rate (WSR), the harmless degree H (Higher is better), and prediction accuracy (Pred Acc).

L	*Method*↓ *Trigger*→	*Noise* *Pred Acc.*	*H*	*WSR*	*Patch* *Pred Acc.*	*H*	*WSR*	*Black-edge* *Pred Acc.*	*H*	*WSR*
32	No WM	91.36	/	/	91.36	/	/	91.36	/	/
	Backdoor	90.97	89.87	1.000	87.56	85.38	1.000	87.92	84.68	1.000
	IW	91.28	91.26	1.000	91.25	91.26	1.000	91.03	91.24	1.000
48	No WM	91.36	/	/	91.36	/	/	91.36	/	/
	Backdoor	90.94	89.38	1.000	89.31	86.49	1.000	89.25	85.22	1.000
	IW	91.31	91.29	1.000	91.14	91.12	1.000	91.11	91.09	1.000
64	No WM	91.36	/	/	91.36	/	/	91.36	/	/
	Backdoor	90.91	89.24	1.000	90.02	87.83	1.000	89.81	87.04	1.000
	IW	91.22	91.19	1.000	90.89	90.68	1.000	90.97	90.73	1.000
128	No WM	91.36	/	/	91.36	/	/	91.36	/	/
	Backdoor	90.89	87.68	1.000	88.47	87.21	1.000	90.03	87.36	1.000
	IW	91.34	91.32	1.000	91.12	90.01	1.000	91.26	90.17	1.000
256	No WM	91.36	/	/	91.36	/	/	91.36	/	/
	Backdoor	90.85	86.33	0.979	90.18	81.65	0.998	90.11	85.79	1.000
	IW	91.26	91.23	1.000	89.63	89.87	1.000	91.29	90.03	1.000

'L' represents the size of the embedded watermark.

Table 4. Performance on ImageNet dataset: watermark success rate (WSR), the harmless degree H (Higher is better), and prediction accuracy (Pred Acc).

L	*Method*↓ *Trigger*→	*Noise* *Pred Acc.*	*H*	*WSR*	*Patch* *Pred Acc.*	*H*	*WSR*	*Black-edge* *Pred Acc.*	*H*	*WSR*
32	No WM	75.81	/	/	75.81	/	/	75.81	/	/
	Backdoor	72.09	71.87	0.813	71.63	71.38	0.806	71.72	71.40	0.811
	IW	75.42	75.39	1.000	75.48	75.41	0.996	75.46	75.40	1.000
64	No WM	75.81	/	/	75.81	/	/	75.81	/	/
	Backdoor	73.25	72.16	0.857	73.31	72.92	0.864	73.43	73.12	0.896
	IW	75.53	75.41	1.000	75.55	75.43	0.998	75.62	75.45	1.000
256	No WM	75.81	/	/	75.81	/	/	75.81	/	/
	Backdoor	73.28	72.31	0.932	73.34	72.99	0.941	73.41	73.08	0.935
	IW	75.64	75.49	1.000	75.67	75.51	1.000	75.68	75.53	0.997

'L' represents the size of the embedded watermark.

$$H = \frac{1}{\left|I_{in} \cup I_{in}^{T}\right|} \sum_{I \in I_{in} \cup I_{in}^{T}} \chi\{f(I;\phi) = g(I)\} \tag{7}$$

where the function $g(I)$ consistently outputs the ground-truth label for I. A larger H means the watermarks have less effect on the utility of the models.

As presented in Table 3 and Table 4, our IW outperforms backdoor-based methods, as reflected in its higher harmlessness degree H. For instance, with a trigger size of 256, our IW achieves an H value

 doi:10.20944/preprints202509.1649.v1

approximately 5.5% higher than that of backdoor-based watermarking techniques, demonstrating its effectiveness, which is comparable to or even surpasses that of baseline backdoor-based approaches.

4.2. Resilience Against Watermark Removal Attacks

Once adversaries acquire the model from external sources, they may employ different strategies to eliminate watermarks or bypass detection. In this section, we assess whether our IW can withstand such attempts. We specifically examine two types of attacks: fine-tuning and model pruning. Fine-tuning involves training the watermarked model on a local benign dataset for a limited number of epochs. In this type of attack, an adversary seeks to eliminate the embedded watermark by fine-tuning the model. On the other hand, model pruning can act as a watermark-removal attack by eliminating neurons associated with the watermark. In this study, we perform pruning by setting to zero the neurons with the smallest l_1 norm. Specifically, the pruning rate represents the fraction of neurons that are removed.

As shown in Figure 4, we firstly test the resilience against fine-tuning attack (left). We fine-tune IW-watermarked models for 30 epochs using the testing set, achieving a WSR exceeding 0.84. These findings highlight the robustness of our IW against fine-tuning attacks. We attribute this mainly to preserving the original labels of watermarked samples during training, which minimizes the impact of fine-tuning compared to backdoor-based approaches. Then, we evaluate the resilience against model-pruning attack (right). As the pruning rate increases, the prediction accuracy of ResNet-18 declines, demonstrating a reduction in the model's effectiveness. Nevertheless, the WSR remains above 0.9. These findings indicate that our IW withstands the model-pruning attack.

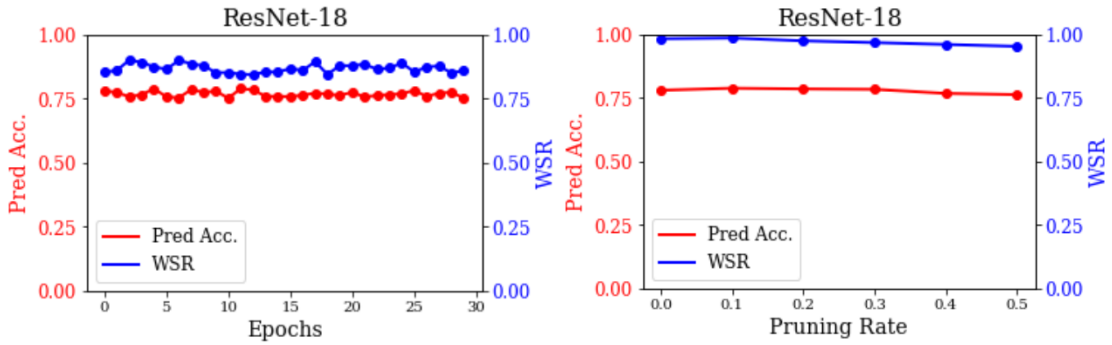

Figure 4. Watermark success rate (WSR) and prediction accuracy of watermarked ResNet-18 against fine-tuning attack (left) and model-pruning attack (right) respectively.

4.3. Latency of Model Data Authentication

We evaluate time latency of BIMW for model data authentication consisting of six stages. It needs three stages to publish model and samples at model owner side: 1) Watermark embedding; 2) Publish model data onto IPFS; 3) Record reference on Blockchain. The user needs three stages to verify model data integrity and model ownership: 4) Query reference from Blockchain; 5) Retrieve model data from IPFS and verify integrity; 6) Watermark extraction. We conducted 50 Monte Carlo test runs for each test scenario and used the averages to measure the results.

Table 5 shows time latency incurred by key stages of BIMW during model authentication process. All test cases are conducted on Jetson Orin Nano platform. The watermark embedding process takes an average of 0.11 s to create interpretable watermarked model on the owner's side. At model users' side, it takes about 0.02 s to extract watermarks by using trigger sample images and verify model's ownership. The numeral results demonstrate efficiency of running the proposed innocuous Model watermarking method under the edge computing environment.

 doi:10.20944/preprints202509.1649.v1

Table 5. Latency of model data authentication process (Seconds).

Stage	1	2	3	4	5	6
Latency	0.11	4.5	1.36	0.53	0.66	0.02

Model Data Authentication Stage: 1: Watermark embedding; 2: Publish model data onto IPFS; 3: Record reference on Blockchain; 4: Query reference from Blockchain; 5: Retrieve model data from IPFS and verify integrity; 6: Watermark Extraction and ownership verification.

We use 10 trigger sample images (about 86 Bytes per figure) and a watermarked model (about 43.7 MB) to evaluate delays incurred by model data distribution and retrieval atop the Blockchain fabric. Due to the large size of watermarked model, it takes about 4.5 s to upload model data to the IPFS network. However, retrieving model data from the IPFS network only introduces small delays (about 0.7 s). The latency of recording reference through smart contracts is greatly impacted by the consensus protocol. It takes about 1.36 s to save reference on test Ganache network. In contrast, querying reference from Blockchain takes much less time (about 0.5 s). We can see fact that publishing data on Blockchain fabric cause more delays (5.86 s) than verification process (1.19 s). However, these overheads only occur once when owners firstly publish and distribute their model data. In sum, our solution brings lower latency during model data retrieval and verification procedures on edge computing platforms.

5. Conclusions

This paper introduces a novel BIMW, a decentralized AI model watermarking framework to guarantees the security and verifiable ownership of model usages under a distributed edge computing environment. To solve issues in widely applied backdoor-based model watermarking approaches, we propose a innocuous model Watermarking method by leveraging interpretable watermarking algorithm and feature impact analysis. The comprehensive experiments demonstrate the effectiveness, robustness, and harmlessness of BIMW compared with backdoor-based model watermarking solutions. Through integration of a Blockchain-based security fabric, BIMW promises to facilitate secure and trustworthy AI model deployment and sharing under distributed edge computing environments.

However, the current prototype of BIMW remains nascent, and several challenges are yet to be addressed. Our ongoing efforts will focus on several directions. While IW has minimal effect on the watermarked model, it remains an intrusive watermarking technique. It might be more beneficial to explore advanced interpretable AI approaches. As edge computing continues to expand, deploying and managing model watermarks at scale across numerous devices presents significant challenges, particularly in terms of scalability and management efficiency, making it essential to ensure their reliable performance across different hardware platforms and devices.

Author Contributions: Conceptualization, X.L. and R.X.; methodology, X.L. and R.X.; software, X.L. and R.X.; validation, X.L. and R.X.; formal analysis, X.L. and R.X.; investigation, X.L. and R.X.; resources, X.L. and R.X.; data curation, X.L. and R.X.; writing—original draft preparation, X.L. and R.X.; writing—review and editing, X.L. and R.X.; visualization, X.L. and R.X.; supervision, R.X.; project administration, R.X.; funding acquisition, R.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no funding.

Data Availability Statement: The original data presented in the study are openly available in the following datasets: [CIFAR-10 datasets] [https://www.cs.toronto.edu/~kriz/cifar.html]; [ImageNet datasets] [https://www.image-net.org/index.php];

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

 doi:10.20944/preprints202509.1649.v1

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
EI	Edge Intelligence
IW	Interpretable Watermarking
IoT	Internet of Things
DL	Deep learning
IP	Intellectual property
FIA	Feature impact analysis
IML	Interpretable machine learning
SHAP	Shapley Additive Explanations
LIME	Local Interpretable Model-agnostic Explanations
CIDs	Content Identifiers
WSR	Watermark success rate

References

1. Liu, X.; Xu, R.; Chen, Y. A Decentralized Digital Watermarking Framework for Secure and Auditable Video Data in Smart Vehicular Networks. *Future Internet* **2024**, *16*.
2. Qu, Q.; Xu, R.; Sun, H.; Chen, Y.; Sarkar, S.; Ray, I. A digital healthcare service architecture for seniors safety monitoring in metaverse. In Proceedings of the 2023 IEEE international conference on metaverse computing, networking and applications (MetaCom). IEEE, 2023, pp. 86–93.
3. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE* **2019**, *107*, 1738–1762.
4. Xu, R.; Nagothu, D.; Chen, Y. AR-Edge: Autonomous and Resilient Edge Computing Architecture for Smart Cities. In *Edge Computing Architecture-Architecture and Applications for Smart Cities*; IntechOpen, 2024.
5. Deng, S.; Zhao, H.; Fang, W.; Yin, J.; Dustdar, S.; Zomaya, A.Y. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet of Things Journal* **2020**, *7*, 7457–7469.
6. Wu, H.; Zhang, Z.; Guan, C.; Wolter, K.; Xu, M. Collaborate edge and cloud computing with distributed deep learning for smart city internet of things. *IEEE Internet of Things Journal* **2020**, *7*, 8099–8110.
7. Li, L.; Ota, K.; Dong, M. Deep learning for smart industry: Efficient manufacture inspection system with fog computing. *IEEE Transactions on Industrial Informatics* **2018**, *14*, 4665–4673.
8. Zhang, Y.; Jia, R.; Pei, H.; Wang, W.; Li, B.; Song, D. The secret revealer: Generative model-inversion attacks against deep neural networks. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 253–261.
9. Tian, Z.; Cui, L.; Liang, J.; Yu, S. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Computing Surveys* **2022**, *55*, 1–35.
10. Liang, Y.; Xiao, J.; Gan, W.; Yu, P.S. Watermarking techniques for large language models: A survey. *arXiv preprint arXiv:2409.00089* **2024**.
11. Wang, R.; Li, H.; Mu, L.; Ren, J.; Guo, S.; Liu, L.; Fang, L.; Chen, J.; Wang, L. Rethinking the vulnerability of dnn watermarking: Are watermarks robust against naturalness-aware perturbations? In Proceedings of the Proceedings of the 30th ACM International Conference on Multimedia, 2022, pp. 1808–1818.
12. Jia, H.; Choquette-Choo, C.A.; Chandrasekaran, V.; Papernot, N. Entangled watermarks as a defense against model extraction. In Proceedings of the 30th USENIX security symposium (USENIX Security 21), 2021, pp. 1937–1954.
13. Yan, Y.; Pan, X.; Zhang, M.; Yang, M. Rethinking {White-Box} watermarks on deep learning models under neural structural obfuscation. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 2347–2364.
14. Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In Proceedings of the 27th USENIX security symposium (USENIX Security 18), 2018, pp. 1615–1631.
15. Shao, S.; Li, Y.; Yao, H.; He, Y.; Qin, Z.; Ren, K. Explanation as a watermark: Towards harmless and multi-bit model ownership verification via watermarking feature attribution. *arXiv preprint arXiv:2405.04825* **2024**.

 doi:10.20944/preprints202509.1649.v1

16. Singh, P.; Devi, K.J.; Thakkar, H.K.; Bilal, M.; Nayyar, A.; Kwak, D. Robust and secure medical image watermarking for edge-enabled e-healthcare. *IEEE Access* **2023**, *11*, 135831–135845.
17. Liu, X.; Xu, R.; Peng, X. BEWSAT: blockchain-enabled watermarking for secure authentication and tamper localization in industrial visual inspection. In Proceedings of the Eighth International Conference on Machine Vision and Applications (ICMVA 2025). SPIE, 2025, Vol. 13734, pp. 54–65.
18. Xu, R.; Liu, X.; Nagothu, D.; Qu, Q.; Chen, Y. Detecting Manipulated Digital Entities Through Real-World Anchors. In Proceedings of the International Conference on Advanced Information Networking and Applications. Springer, 2025, pp. 450–461.
19. Saha, A.; Subramanya, A.; Pirsiavash, H. Hidden trigger backdoor attacks. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2020, Vol. 34, pp. 11957–11965.
20. Li, E.; Zhou, Z.; Chen, X. Edge intelligence: On-demand deep learning model co-inference with device-edge synergy. In Proceedings of the Proceedings of the 2018 workshop on mobile edge communications, 2018, pp. 31–36.
21. Liu, X.; Xu, R.; Zhao, C. AGFI-GAN: An Attention-Guided and Feature-Integrated Watermarking Model Based on Generative Adversarial Network Framework for Secure and Auditable Medical Imaging Application. *Electronics* **2024**, *14*, 86.
22. Boenisch, F. A systematic review on model watermarking for neural networks. *Frontiers in big Data* **2021**, *4*, 729663.
23. Pan, X.; Zhang, M.; Yan, Y.; Wang, Y.; Yang, M. Cracking white-box dnn watermarks via invariant neuron transforms. In Proceedings of the Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 1783–1794.
24. Li, Y.; Zhu, M.; Yang, X.; Jiang, Y.; Wei, T.; Xia, S.T. Black-box dataset ownership verification via backdoor watermarking. *IEEE Transactions on Information Forensics and Security* **2023**, *18*, 2318–2332.
25. Battah, A.; Madine, M.; Yaqoob, I.; Salah, K.; Hasan, H.R.; Jayaraman, R. Blockchain and NFTs for trusted ownership, trading, and access of AI models. *IEEE Access* **2022**, *10*, 112230–112249.
26. Molnar, C. *Interpretable machine learning*; Lulu. com, 2020.
27. Murdoch, W.J.; Singh, C.; Kumbier, K.; Abbasi-Asl, R.; Yu, B. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences* **2019**, *116*, 22071–22080.
28. Ribeiro, M.T.; Singh, S.; Guestrin, C. " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.
29. Garreau, D.; Luxburg, U. Explaining the explainer: A first theoretical analysis of LIME. In Proceedings of the International conference on artificial intelligence and statistics. PMLR, 2020, pp. 1287–1296.
30. Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561* **2014**.
31. Jetson Orin Nano Super Developer Kit. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit/. Accessed on March 2025.
32. Solidity. https://docs.soliditylang.org/en/v0.8.13/. Accessed on March 2025.
33. Ganache. [Online]. Available: https://archive.trufflesuite.com/ganache/. Accessed on March 2025.
34. Solidity. https://archive.trufflesuite.com/docs/truffle/. Accessed on March 2025.
35. IPFS Docs. [Online]. Available: https://docs.ipfs.tech/. Accessed on March 2025.