symmetry

Article

Clean-Label Backdoor Watermarking for Dataset Copyright Protection via Trigger Optimization

Weitong Chen [1,2], Gaoyang Wei [1], Xin Xu [1,2,*], Yanyan Xu [3], Haibo Peng [4] and Yingchen She [4]

[1] School of Information Engineering, Yangzhou University, Yangzhou 225127, China; wtchen@yzu.edu.cn (W.C.); 211302222@yzu.edu.cn (G.W.)
[2] Jiangsu Province Engineering Research Center of Knowledge Management and Intelligent Service, Yangzhou 225127, China
[3] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China; xuyy@whu.edu.cn
[4] The Third Surveying and Mapping Institute of Hunan Province, Changsha 410018, China; penghaibo_hunan@outlook.com (H.P.); sheyingchen_hunan@outlook.com (Y.S.)
* Correspondence: mx120230591@stu.yzu.edu.cn

Abstract: High-quality datasets are essential for training high-performance models, while the process of collection, cleaning, and labeling is costly. As a result, datasets are considered valuable intellectual property. However, when security mechanisms are symmetry-breaking, creating exploitable vulnerabilities, unauthorized use or data leakage can infringe on the copyright of dataset owners. In this study, we design a method to mount clean-label dataset watermarking based on trigger optimization, aiming to protect the copyright of the dataset from infringement. We first perform iterative optimization of the trigger based on a surrogate model, with targets class samples guiding the updates. The process ensures that the optimized triggers contain robust feature representations of the watermark target class. A watermarked dataset is obtained by embedding optimized triggers into randomly selected samples from the watermark target class. If an adversary trains a model with the watermarked dataset, our watermark will manipulate the model's output. By observing the output of the suspect model on samples with triggers, it can be determined whether the model was trained on the watermarked dataset. The experimental results demonstrate that the proposed method exhibits high imperceptibility and strong robustness against pruning and fine-tuning attacks. Compared to existing methods, the proposed method significantly improves effectiveness at very low watermarking rates.

Keywords: dataset watermarking; trigger optimization; clean label; copyright protection; backdoor

check for updates

Citation: Chen, W.; Wei, G.; Xu, X.; Xu, Y.; Peng, H.; She, Y. Clean-Label Backdoor Watermarking for Dataset Copyright Protection via Trigger Optimization. *Symmetry* **2024**, *16*, 1494. https://doi.org/10.3390/sym16111494

Academic Editor: Jie Yang

Received: 13 October 2024
Revised: 28 October 2024
Accepted: 6 November 2024
Published: 8 November 2024

1. Introduction

Nowadays, companies use deep models to provide intelligent services or enhance product performance [1–3]. High-quality datasets are essential for training high-performance models [4,5]. Creating datasets requires extensive data collection, processing, and labeling, which is both costly and time-consuming [6]. A well-curated proprietary dataset may have an extremely high application value in certain fields and may even be irreplaceable [7]. For dataset owners, these datasets have unique commercial values, which are their intellectual property. Symmetry in dataset security ensures that the protection mechanisms, such as watermarking, create a balanced defense against unauthorized use or theft. However, symmetry-breaking occurs when vulnerabilities are exploited asymmetrically, such as in targeted attacks, where specific datasets in data infrastructure will be the target of hackers. These datasets are facing the risk of being stolen by competitors or misused by attackers. As a result, the copyright protection of datasets has become an increasingly important research topic in recent years.

In the dataset copyright protection scenario, the purpose of attackers stealing the dataset is to train their commercial model for profit [8]. Typically, the models are deployed in the cloud, and only the inference APIs are released to users [9,10]. The dataset owner needs to determine whether the suspect model has been trained using their dataset and can only gather evidence through the model's API. This makes the method of evidence collection in dataset copyright protection scenarios fundamentally different from that of traditional image data copyright protection. Currently, there are several techniques for protecting data property rights, such as differential privacy [11], digital image watermarking [12], and data encryption [13]. However, these methods are not well suited for dataset copyright protection because unauthorized users typically release only their trained models and not the original training data. Additionally, dataset-watermarking-based copyright protection requires verifying the watermark through the model's output [14], which differs from traditional methods that rely on direct access to the data. Therefore, traditional image watermarking methods are not capable of dataset copyright protection scenarios.

Research on dataset watermarking is lacking. Inspired by deep learning backdoor attacks [15–19], Li et al. [20] first propose a dataset-watermarking method. This method employs a poison-label backdoor attack to embed watermarks by adding watermark triggers, such as pixel patterns [16], to a portion of original samples in the dataset to generate poisoned samples. Meanwhile, these samples are labeled as a predefined target class, causing a mismatch between the semantic information of the samples and their labels. When a model is trained using such a dataset, it learns the mapping between the watermark trigger and the predefined target class [21]. When verifying whether a suspect model has used the watermarked dataset during training, samples containing the watermark trigger are fed into the model, and the rate of these samples predicted as the predefined class is observed for watermark verification. However, the poison-label-based dataset-watermarking methods are easily detected through manual review, as the samples are obviously wrong-labeled. Attackers can remove these obvious trigger samples before training the model, rendering the watermarking algorithm ineffective.

To improve the imperceptibility of dataset watermarking, Tang et al. [22] propose a dataset-watermarking method based on the mechanism of clean-label backdoor attacks [23,24]. Unlike poison-label attacks [16] or label-flipping attacks [25], this clean-label method embeds watermark triggers into samples of a predefined class without altering their labels, while still achieving a mapping between the watermark triggers and the predefined target class. This is critical for dataset watermarking, as non-clean-label backdoor methods are highly susceptible to detection during human review. Additionally, Li et al. [26] implement a clean-label dataset watermarking through untargeted backdoor, where the watermark trigger does not point to a predefined class. As a result, watermark verification requires knowledge of the predicted probability of each class and is conducted through a hypothesis test. This verification method requires full access to the suspect model and cannot be performed solely based on the model's API, which limits the practicality of the algorithm. Although clean-label-based algorithms can effectively achieve dataset copyright protection while maintaining imperceptibility, there still are limitations. Since the trigger samples all come from the watermark target class, the model may confuse the trigger features with the features of the target class during training. On the one hand, this leads to a decrease in prediction accuracy for clean samples of the target class. On the other hand, it means that clean-label algorithms require a higher watermarking rate than poison-label algorithms to ensure watermark effectiveness.

To address these issues, we propose a clean-label dataset-watermarking method based on trigger optimization. Specifically, the trigger optimization aims to generate watermark triggers that contain robust feature representations of the target class. We first generate a surrogate model by training on the public out-of-distribution (OOD) dataset and fine tuning it with target class samples. Then, we freeze the parameters of the surrogate model and iteratively update the input trigger pattern to make it closer to the specific feature representations of the target class. Simultaneously, a trigger optimization constraint is employed to ensure the trigger is imperceptible. Replace the original samples with the samples embedded with optimized triggers to achieve watermark embedding and obtain the watermarked dataset. If an unauthorized user utilizes this dataset as part of the training set to train their third-party model, the model will be backdoored. During watermark verification, only access to the suspect model's API is required. The watermark is verified by inputting trigger samples and using the Wilcoxon test.

The main contributions are summarized as follows:

- We propose a dataset copyright protection method based on clean-label backdoor watermarking and trigger optimization, which can be used to verify whether a suspect model has used our dataset during training.
- We construct a surrogate model and optimize the trigger through iterative guided updates using target class samples. This allows the optimized triggers to have robust feature representations of the target class, making them harmless and more effective.
- We utilize a clean-label watermark embedding setting to ensure that the semantic information of the trigger samples aligns with their labels, making the watermark information imperceptible.
- Extensive experiments have demonstrated the effectiveness, high imperceptibility, and strong robustness of the proposed method. It can maintain effectiveness at a considerably lower watermarking rate compared with existing methods.

The rest of this paper is organized as follows. In Section 2, we introduce the preliminaries and objectives of the proposed method. In Section 3, we present the proposed method in detail. In Section 4, we conduct experiments to evaluate the performance of the proposed method and discuss the results of the experiments. Finally, we conclude this paper in Section 5.

2. Preliminaries and Objectives

In this section, we will discuss the mechanism of backdoor-based dataset watermarking and the goals that need to be achieved to ensure effective dataset watermarking.

2.1. The Mechanism of Backdoor-Based Dataset Watermarking

We consider a dataset-watermarking scenario in which the defender and the adversary act as follows. Suppose that $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$, where $x_i \in \{0, 1, 2, \cdots, 255\}^{C\times W\times H}$ and $y_i \in \{1, 2, 3, \cdots, K\}$ represent the original dataset with K classes and N samples; here, x_i is the original sample and y_i is the corresponding label. Let $G_w : x \to x^*$ be the trigger embedding function, which embeds watermark triggers δ into a clean sample x, where x^* is the watermarked sample. δ refers to a set of image patterns, such as squares, lines, or slight perturbations generated by a specific algorithm [24]. A certain proportion of samples are selected from $\mathcal{D}$, defined as $\mathcal{D}_s$. In poison-label dataset watermarking, $\mathcal{D}_s$ is composed of samples selected from each class. In clean-label dataset watermarking, $\mathcal{D}_s$ consists only of samples from the target class. Embed δ into $\mathcal{D}_s$ to generate $\mathcal{D}_s^*$:

$$\mathcal{D}_s^* = \{(G_w(x_i), t)\}, \text{ where } (x_i, y_i) \in \mathcal{D}_s \tag{1}$$

where t is the watermark target label.

The watermarking rate $\gamma = \frac{||\mathcal{D}_s||}{||\mathcal{D}||}$ is defined as the proportion of watermarked samples in the entire dataset. The smaller γ is, the more challenging it is to detect the watermark. The final watermarked dataset $\mathcal{D}_w = \mathcal{D}_s^* \cup (\mathcal{D} \setminus \mathcal{D}_s)$ is a combination of watermarked samples and the remaining original samples. The defender releases only the watermarked dataset while keeping the original dataset confidential. The adversary uses the watermarked dataset without authorization to train their third-party model, which will be backdoored. As shown in Figure 1, a white square pattern is embedded into the samples as the trigger. During the training stage, the presence of triggered samples manipulates the adversary's model, embedding the watermark into the model's behavior. In the inference stage, when the backdoored model receives inputs containing the watermark trigger, it manipulates the output to predict the predefined watermark target class t set by the defender.

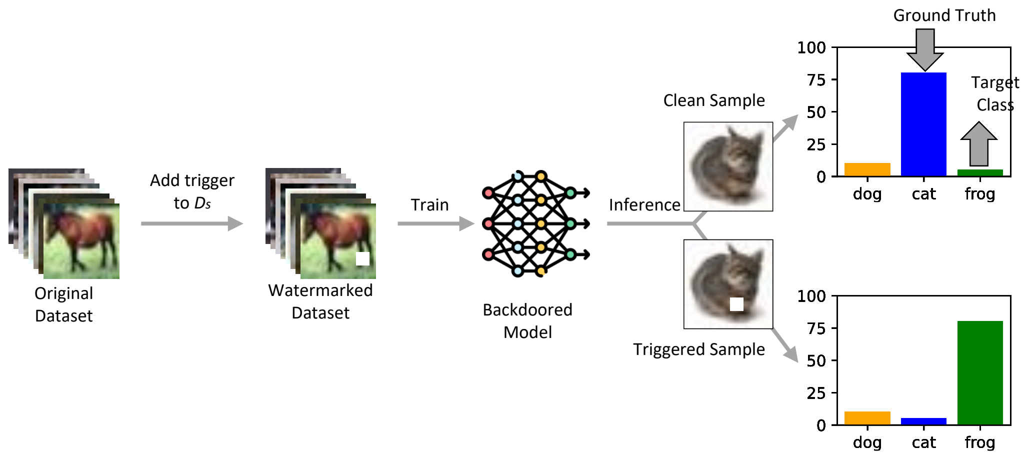


Figure 1. The system model of dataset watermarking.

2.2. Design Goals

In this section, we clarify our design goals to evaluate our method in subsequent chapters. Our design goals are as follows:

- Effectiveness: The verification algorithm should accurately detect whether the target model has used the protected dataset.
- Imperceptibility: Watermarking should not be noticeable to those who might steal the dataset.
- Harmlessness: The watermarking algorithm should not affect the performance of the model regarding primary tasks.
- Robustness: Watermarks should be resistant to potential attacks, such as fine tuning and model pruning.

2.2.1. Effectiveness

In this context, effectiveness refers to the watermark verification algorithm's ability to accurately detect whether the target model has utilized the protected dataset. This process can be concisely described as follows.

$$\text{verify}(f_b, x^*) = \text{True} \tag{2}$$

where f_b represents the model trained with the watermarked dataset. verify$(\cdot)$ is the watermark verification algorithm, which returns either True or False for a successful verification or not.

2.2.2. Imperceptibility

Inperceptibility refers to the characteristic when the watermarked dataset does not exhibit noticeable differences compared to the original datasets. The goal is to prevent our adversaries from easily distinguishing between samples containing watermark triggers and original samples. If adversaries can easily differentiate between the two, they could circumvent backdoor watermarking by excluding the watermarked samples during training [23]. The imperceptibility of the watermarking algorithm is primarily influenced by three factors:

1. Visual Quality: This factor indicates the difference in visual quality between watermarked and original samples, which is denoted by the following formula.

$$\text{diff}(x^*, x) \leqslant \varepsilon \tag{3}$$

where diff is the difference evaluation function and ε is the threshold for imperceptibility.

2. Label Alignment: This factor indicates whether the labels of watermarked samples are consistent with their ground truth. Otherwise, it would be easy to manually detect that the samples have been mislabeled.

$$y = t, \quad \forall (x^*, t) \in \mathcal{D}_w, (x, y) \in \mathcal{D} \tag{4}$$

3. Watermarking Rate: This factor γ indicates the proportion of watermarked samples relative to the total number of samples in the dataset. The lower the γ, the harder it is for a watermark to be detected by adversaries.

2.2.3. Harmlessness

Harmlessness measures the impact of the watermark on the performance of the main task. Our goal is to ensure that watermarking does not significantly affect the accuracy of the target model, which is defined by

$$f_c(x) \approx f_b(x) \tag{5}$$

where $f_c(\cdot)$ represents the clean model. The higher the harmlessness, the more similar the results between the two models.

2.2.4. Robustness

Robustness refers to the resistance and stability of our watermark against various attacks or processing operations. It indicates that the watermark can be reliably detected even if the target model is subjected to attacks. We can define it as follows:

$$\forall\, \text{verify}(f_{b'}, x^*) = \text{True} \tag{6}$$

where $f_{b'}$ represents the target model after attacks such as fine tuning or pruning.

3. Proposed Methods

As shown in Figure 2, the procedure consists of four steps: (a) trigger optimization, (b) dataset watermark embedding, (c) adversary model manipulation, and (d) watermark verification.

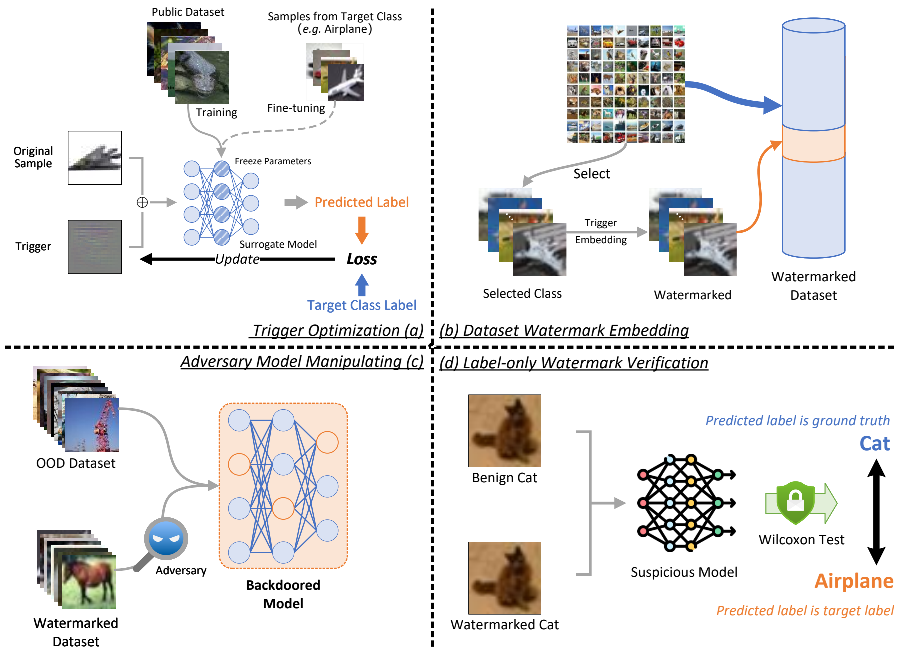


Figure 2. The workflow of the proposed method. (**a**) Trigger Optimization: Iterative optimization of triggers through a surrogate model. (**b**) Dataset Watermark Embedding: Embedding triggers into

randomly selected samples from the target class. (**c**) Adversary Model Manipulation: The adversary trains the model using the watermarked dataset, and the watermark will manipulate the attacker's model for backdoor implantation. (**d**) Label-only Watermark Verification: Verify whether a suspicious model was trained on the watermarked dataset.

3.1. Trigger Optimization

In existing clean-label dataset watermarking, a higher γ is required compared to the poison-label methods to ensure effectiveness due to the key limitation of clean-label dataset watermarking, which is the mismatch of the watermark trigger and the target class. We frame the generation of more effective triggers as an optimization problem.

Specifically, this optimization problem is defined as finding an optimized trigger pattern δ^* such that for each sample (x, t) in the target class, the prediction loss $\mathcal{L}(f_c(x+\delta), t)$ is minimized. Here, f_c represents a model trained on the original dataset. Therefore, δ^* should capture the most stable and representative features of the target class t, so that adding it to any input maximizes the likelihood of it being predicted as t. Additionally, the number of target samples in the original dataset may not be enough to create a robust feature extraction model in practice. Hence, we adopt a surrogate model f_s, which is trained using a two-stage approach. It is first trained on a public OOD dataset to capture general class features and then fine-tuned on a small set of target class samples to learn specific features unique to the target class. This helps the model build robust and generalizable patterns, which are then used to guide the trigger optimization process. The public OOD dataset could be collected from the Internet or open source datasets. Then, the surrogate model is used to guide the optimization of the trigger pattern. Therefore, the optimization problem is now formulated as follows:

$$\delta^* = \underset{\delta \in \Delta}{\operatorname{argmin}} \sum_{(x,t) \in \mathcal{D}_w} \mathcal{L}(f_s(x+\delta), t) \tag{7}$$

where Δ represents the allowable set of trigger patterns.

A step-by-step explanation of the trigger optimization process is provided below, along with the pseudo-code illustrated in Algorithm 1.

Algorithm 1: Trigger optimization algorithm.

```
Input: $f_s$: Surrogate model
       $\mathcal{D}_t$: Target class samples
       $\Delta$: Trigger optimization constraint
       $\mathcal{T}$: Total iterations
Output: $\delta_{\mathcal{T}}$: Optimized trigger
/* 1.Trigger initialization */
1 $\delta_0 \leftarrow \mathbf{0}^{1\times d}$;
2 for $i \in (1, \mathcal{T}-1)$ do
      /* 2.  Trigger updating */
3     $\delta_{i+1} \leftarrow \delta_i - \sum_{(x,t)\in\mathcal{D}_t} \nabla_\delta \mathcal{L}(f_s(x+\delta_i), t)$;
      /* 3.  Trigger constraint */
4     $\delta_{i+1} \leftarrow \mathrm{Proj}_\Delta(\delta_{i+1})$ ;
5 end
6 return $\delta_{\mathcal{T}}$
```

1. Trigger Initialization: Initialize a trigger δ_0, typically set to a zero vector or small random noise.
2. Iterative Update: Freeze the parameters of the surrogate model f_s and input $x_i + \delta$. Perform $\mathcal{T}$ rounds of iterations, updating δ in each iteration based on the target class

data (x, t). In each iteration, the gradient of the loss function $\mathcal{L}(f_s(x+\delta), t)$ with respect to the current trigger δ_i is calculated.

3. Gradient Descent: After averaging these gradients, update the trigger using gradient descent based on the update formula $\delta_{i+1} = \delta_i - \alpha \nabla_\delta \mathcal{L}$, where α is the learning rate.
4. Trigger Optimization Constraint: To ensure the imperceptibility of the trigger, δ is projected onto the allowed set Δ after each update. Δ is defined as ℓ^∞-norm, i.e., $\Delta = \{\delta \| \|\delta\|_\infty \leq \varepsilon\}$. This step can be performed by simply clipping each dimension of δ to $[-\varepsilon, +\varepsilon]$.
5. Optimization Complete: After the iteration is complete, output the optimized trigger $\delta_{\mathcal{T}}$ of x_i.

3.2. Dataset Watermark Embedding

In our study, we use the clean-label dataset-watermarking setup, where only a portion of the target class samples are embedded with the trigger. Given a specified watermarking rate γ, randomly select the corresponding number of samples from the watermark target class in the original dataset $\mathcal{D}$. The set of selected samples is denoted as $\mathcal{D}_s$. Initialize a trigger for each selected sample individually and perform trigger optimization. Therefore, in our method, use the trigger embedding function $G_w(x_i) = x_i + \delta_{\mathcal{T}}$. The watermark embedding process is described as follows:

$$\mathcal{D}_w = \begin{cases} (G_w(x_i), y_i) & \text{if } (x_i, y_i) \in \mathcal{D}_s \\ (x_i, y_i) & \text{otherwise} \end{cases}, \quad (x_i, y_i) \in \mathcal{D} \tag{8}$$

Since the optimized trigger is imperceptible and the labels have not been modified, it becomes difficult to distinguish the samples with triggers from normal samples in terms of labels and surface features.

3.3. Adversary Model Manipulating

Assume that after illegally obtaining our watermarked dataset $\mathcal{D}_w$, the adversary uses it along with an additional dataset obtained through other means to train their own model. The model will be manipulated by $\mathcal{D}_w$. It can be considered a backdoored model f_b.

3.4. Watermark Verification

We propose a verification method based on the hypothesis test and the Wilcoxon signed-rank test [14] to evaluate dataset ownership. This method provides a non-parametric approach to detect significant differences in model behavior, which may indicate unauthorized use of a protected dataset.

Specifically, the defender first selects m number of samples from the protected dataset to build $\mathcal{D}_v = \{(x_i, y_i) | y_i \neq t\}_{i=1}^m$, where these samples have labels different from t. Then, they conduct trigger embedding $x_i^* = G_w(x_i)$ to form the watermark verification set $\mathcal{D}_v^* = \{x_i^*\}$. Next, the samples in $\mathcal{D}_v^*$ are sequentially input into the suspicious model f_θ for prediction, and the predicted labels $f_\theta(x^*)$ are stored in set Y in order.

The Wilcoxon signed-rank test is performed to calculate the p-value in the following steps.

1. Calculate the Differences: For each pair of data points, calculate the difference as follows:

$$d_i = \mathcal{Y}_i - t \tag{9}$$

Discard any data points where $d_i = 0$.

2. Rank the Absolute Differences: Rank the absolute values of the non-zero differences, assigning ranks R_i to the absolute values $|d_i|$, starting from 1.

3. Assign Signs to the Ranks: The rank R_i is given a positive sign if $d_i > 0$ and a negative sign if $d_i < 0$.

4. Sum the Ranks: Let W_+ be the sum of the positive ranks and W_- be the sum of the negative ranks.

$$W_+ = \sum_{d_i>0} R_i, \quad W_- = \sum_{d_i<0} R_i \tag{10}$$

5. Test Statistic: The Wilcoxon test statistic is defined as the smaller value of W_+ and W_-:

$$W = \min(W_+, W_-) \tag{11}$$

6. p-value: The p-value is calculated by comparing W to its expected distribution under the null hypothesis H_0. For large samples ($n > 20$), a normal approximation can be used, with a mean and standard deviation given by

$$\mu_W = \frac{n(n+1)}{4}, \quad \sigma_W = \sqrt{\frac{n(n+1)(2n+1)}{24}} \tag{12}$$

where the Z-score is computed as

$$Z = \frac{W - \mu_W}{\sigma_W} \tag{13}$$

The p-value is then calculated from the Z-score using the standard normal distribution φ:

$$p\text{-value} = 2 \times (1 - \varphi(|Z|)) \tag{14}$$

If the p-value is less than the significance level 0.05, we determine that the model was trained on the protected dataset. The watermark verification process is summarized in Algorithm 2.

Algorithm 2: Watermark verification algorithm.

```
Input:  $f_\theta$: Suspicious model
        $G_w$: Trigger embedding function
        $\mathcal{D}_v$: Random selected samples
        $t$: The target label
Output: $p$-value
  /* 1.  Generate watermark verification set */
1 $\mathcal{D}_v^* \leftarrow \{x_i^* = G_w(x_i) | x_i \in \mathcal{D}_v\}$;
  /* 2.  Record the prediction results of watermark verification set */
2 $\mathcal{Y} \leftarrow \{f_\theta(x_i^*) | x_i^* \in \mathcal{D}_v^*\}$;
  /* 3.  Perform Wilcoxon signed-rank test */
3 $p$-value ← WILCONXON($\mathcal{Y}, t$);
4 return $p$-value
```

4. Experiments and Analysis

4.1. General Set-Ups

4.1.1. Datasets

Our evaluation covered three datasets commonly used for image classification tasks: CIFAR-10 [27], Tiny-ImageNet [28], and PubFig [29]. The public OOD datasets used for training the corresponding surrogate models are Tiny-ImageNet [28], CIFAR-100 [27], and CelebA [30], respectively.

4.1.2. Models and Parameters

We employ ResNet-18 [31] as the surrogate model and target model across our experiments. The watermarking rate γ is set to 0.05% for all datasets, which means that only 25/50,000, 50/100,000, and 6/12,454 samples of CIFAR-10, Tiny-ImageNet, and PubFig, respectively, are embedded with triggers. Additionally, following the guidelines from [23,32], we set ε of ℓ^∞-norm to 16/255.

4.1.3. Hardware Platform and Execution Time

The proposed method was implemented in PyTorch. The experiments were carried out on a single workstation. The hardware setup included an Intel Core i9 processor, 64 GB of memory, and an NVIDIA GeForce RTX 4090 GPU. The execution time of the proposed method on the above-mentioned datasets is summarized in Table 1. The execution time of surrogate model training primarily depends on the size of the public OOD dataset. The training times on Tiny-ImageNet, CIFAR-100, and CelebA are 5, 12, and 35 min, respectively. The execution efficiency of trigger optimization is high, with the average execution time per sample being 16.7, 23.8, and 21.6 ms on CIFAR-10, Tiny-ImageNet, and PubFig, respectively. When the watermarking rates are 0.05% and 0.1%, the execution times for dataset watermarking are all within 3000 ms.

Table 1. Execution time of watermarking process.

	CIFAR-10 (Tiny-ImageNet OOD)	**Tiny-ImageNet (CIFAR-100 OOD)**	**PubFig (CelebA OOD)**
Surrogate model training time	5 min	12 min	35 min
Average trigger optimization time	16.7 ms	23.8 ms	21.6 ms
Watermarking time ($\gamma = 0.05\%$)	486.5 ms	1437.6 ms	149.6 ms
Watermarking time ($\gamma = 0.1\%$)	967.8 ms	2988.7 ms	306.1 ms

4.2. Evaluation Metrics

For dataset watermarking, the main performance metrics are as follows:

1. WSR (Watermark Success Rate): WSR refers to the proportion of samples in the watermark verification set that are successfully predicted as the target class by f_b. It is widely used to evaluate the effectiveness of watermarking in protecting intellectual property. A higher WSR indicates better robustness of the watermark, enabling more accurate recognition and verification of the watermark information.
2. MA (Model Accuracy): MA refers to the model's prediction accuracy on the original task. By evaluating MA, we can assess the impact of watermarking on the model's performance. Considering the scenario where authorized users train models using watermarked datasets, dataset watermarking should be harmless to the model. A higher MA indicates that the watermark has a lower impact on the overall performance of the model.
3. TA (Target Accuracy): TA refers to the model's prediction accuracy on clean samples from the target class. Since clean-label dataset-watermarking methods embed watermark triggers only in samples of the target class, the accuracy of the target class may be more significantly affected by the watermark compared to other classes.

In summary, an excellent dataset-watermarking algorithm should achieve a high WSR, MA, and TA.

4.3. Performance Analysis

4.3.1. Comparison on Effectiveness

We compared the proposed method with baseline methods based on backdoor attacks, such as BadNets [16], Blend [15], HTBA [32], LCBA [23], and existing dataset-watermarking methods, such as UBW [26], including both UBW-P and UBW-C. Among these, BadNets, Blend, and UBW-P are based on poison-label methods, while the others are based on clean-label methods. For backdoor attack methods, we utilize the value of the attack success rate as the WSR. Additionally, LCBA offers two approaches, one based on GANs and the other on adversarial perturbations. In this paper, we only make a comparison with the latter, as it is reported to be much more effective than the former in the literature.

For each dataset, we randomly selected samples according to the watermarking rate of 0.05% and embedded triggers to form the watermark verification set. We repeated the random sampling three times for separate experiments and calculated the average of the three results. The experimental results are shown in Table 2, where the bold font indicates the best performance.

Table 2. Experimental results on effectiveness with different methods.

		Benign	**BadNets** [16]	**Blend** [15]	**HTBA** [32]	**LCBA** [23]	**UBW-P** [26]	**UBW-C** [26]	**Ours**
CIFAR-10	p-value		8.9×10^{-38}	1.00	1.00	1.00	6.33×10^{-146}	4.82×10^{-97}	$\mathbf{3.34 \times 10^{-228}}$
	WSR%		88.76	77.79	5.28	4.09	89.98	84.32	**99.03**
	MA%	95.62	95.26	94.66	95.13	**95.57**	90.59	87.21	95.20
	TA%	93.62	93.64	94.08	93.32	93.38	89.21	85.53	**94.10**
Tiny-ImageNet	p-value		1.00	1.00	1.00	1.00	1.00	1.00	$\mathbf{8.38 \times 10^{-236}}$
	WSR%		0.80	0.50	2.16	1.11	1.95	0.11	**85.81**
	MA%	63.22	**65.10**	64.59	64.08	65.08	63.94	63.31	64.65
	TA%	69.11	**70.36**	68.13	67.84	68.20	67.69	67.80	70.00
PubFig	p-value		1.00	1.00	1.00	1.00	1.00	1.00	$\mathbf{5.17 \times 10^{-9}}$
	WSR%		0.00	30.96	0.00	0.85	1.21	2.11	**99.89**
	MA%	93.64	93.47	93.15	**93.85**	93.54	93.04	93.07	93.28
	TA%	95.96	92.21	96.53	97.36	97.46	**97.91**	96.66	95.62

It is worth noting that our experimental setup uses a much lower watermark rate than what is commonly explored in the existing literature, where most studies employ watermark or poisoning rates between 5% and 20%. Because of this, neither the MA nor the TA of all methods showed a significant decline. Based on this result, our method is harmless to model performance. However, most of the comparison methods are ineffective at such a low watermark rate. Only BadNets, UBW-P, UBW-C, and our method remained effective on CIFAR-10, while on Tiny-ImageNet and PubFig, only our method maintained its effectiveness. In all experiments, the proposed method achieved the lowest p-value and the highest WSR. Our method achieved the best performance regarding effectiveness.

4.3.2. Comparison on Imperceptibility

In this section, we assess the stealthiness of our dataset-watermarking framework in comparison to other baselines. We used the Learned Perceptual Image Patch Similarity (LPIPS) [33], which can better account for many nuances of human perception compared to commonly used metrics, like PSNR and SSIM, along with Pixel-level Similarity (ℓ^∞-norm) as evaluation metrics to assess imperceptibility. Lower values of LPIPS and ℓ^∞-norm indicate better imperceptibility and visual quality.

Examples are illustrated in Figure 3. The difference between the sample with the embedded trigger and the original sample is minimal. Additionally, our trigger demonstrates better visual imperceptibility. The statistical experimental results for different methods are shown in Table 3. There, it is shown that, compared to existing methods, the average LPIPS value of our method is the smallest across all three datasets. Therefore, the proposed method exhibits better imperceptibility.

Table 3. Average LPIPS values for different methods.

	BadNets	**LCBA**	**UBW-P**	**UBW-C**	**Ours**
CIFAR-10	0.3833	0.0060	0.3834	0.0008	0.0067
Tiny-ImageNet	0.0239	0.0349	0.0255	0.0443	0.0215
PubFig	0.2401	0.2702	0.2396	0.3155	0.2285

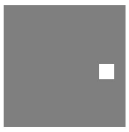
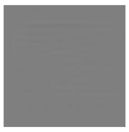
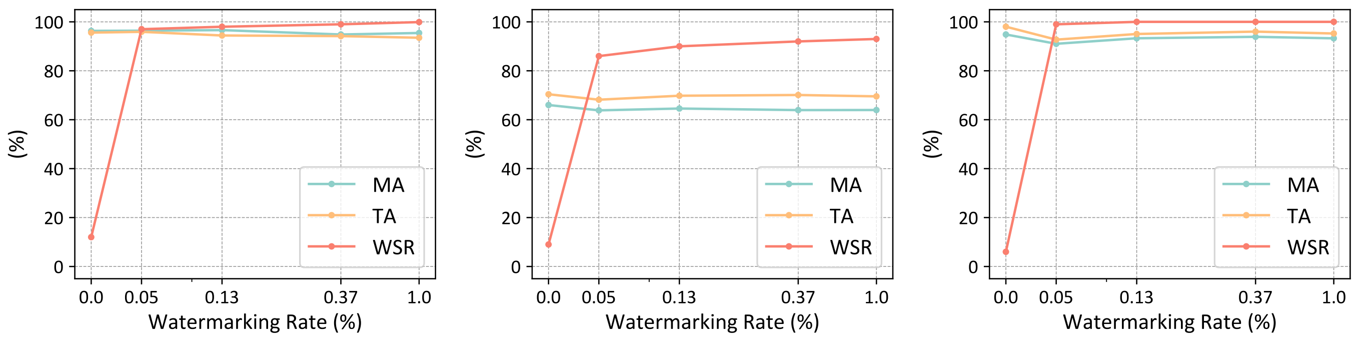
	BadNets	**LCBA**	**UBW-P**	**UBW-C**	**Ours**
Label alignment	poison label	clean label	poison label	clean label	clean label
Samples					
Triggers					
ℓ^∞ **norm**	255/255	16/255	255/255	16/255	16/255
LPIPS	0.0443	0.0364	0.3243	0.0428	0.0203

Figure 3. Example of samples with embedded triggers.

4.3.3. Comparison on Watermarking Rate

Considering that an adversary may train the model using both a watermarked dataset and other datasets, this implies that adding other datasets will reduce the overall watermarking rate. Therefore, analyzing the performance under different watermarking rates is of practical significance. In this section, we investigate the impact of different watermarking rates on the WSR, MA, and TA, and compare the proposed method with the baselines.

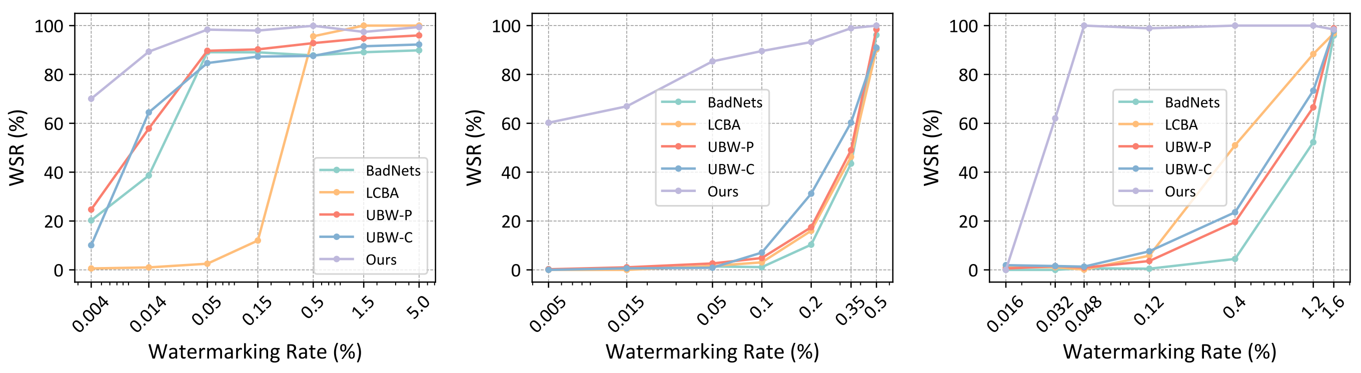
We conducted experiments with a watermarking rate from 0.05% to 1% and recorded the changes in the WSR, MA, and TA. The results are shown in Figure 4. In particular, the watermarking rate of 0% indicates the original sample. As the watermarking rate increases, there is no significant decline in the MA and TA compared with the original samples. Our method can maintain a high WSR across different watermarking rates while preserving harmlessness in terms of the MA and TA, demonstrating that is has a minimal impact on the dataset's utility and the performance of models trained on it.



(**a**) CIFAR-10 (**b**) Tiny ImageNet (**c**) PubFig

Figure 4. Impact of watermarking rate on MA, TA, and WSR.

We further compared our method with the baseline methods of watermarking rates ranging from 0.004% to 5%. The results are shown in Figure 5. It can be found that as the watermarking rate increases, the WSR also increases. In the experiments on CIFAR-10, with a watermarking rate of 0.004%, where only two samples are embedded with triggers, our WSR still exceeded 70%, while the WSR of other compared algorithms stayed below 25%. In the experiments on Tiny-ImageNet and PubFig, our WSR is significantly higher than that of the compared algorithms at low watermarking rates. Our method demonstrated the best performance across all datasets and different watermarking rates.



(**a**) CIFAR-10 (**b**) Tiny ImageNet (**c**) PubFig

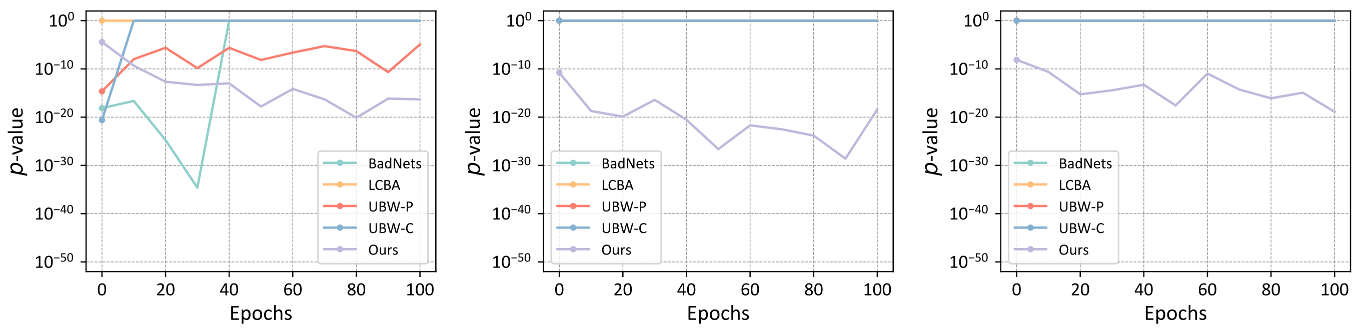
Figure 5. Comparison of WSR with different watermarking rates.

4.3.4. Comparison on Robustness

We consider that the adversary's model may undergo pruning or fine tuning before being released. In this section, we examine the robustness of the proposed dataset-watermarking method against model fine tuning and pruning attacks. We fine-tuned and pruned the model trained on the watermark dataset. Then, we performed watermark verification on the processed model and calculated the WSR.

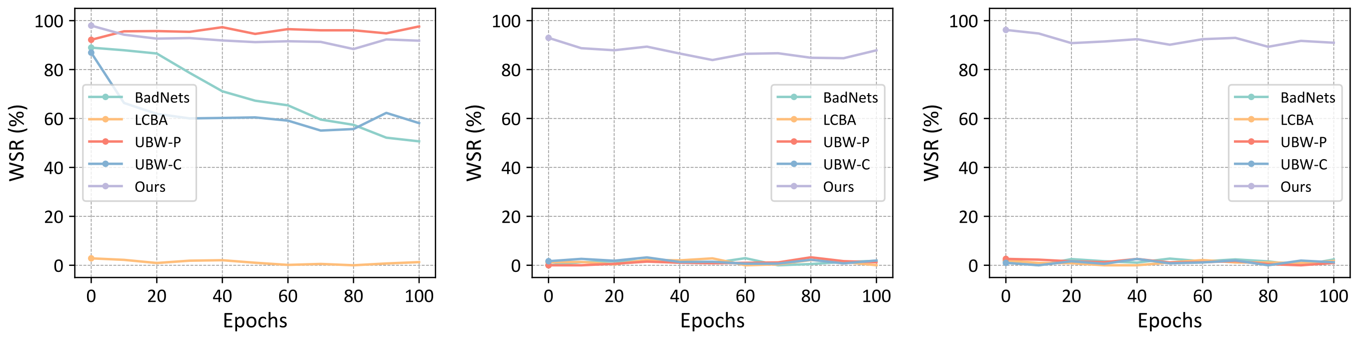
The comparison of robustness against fine tuning attacks is shown in Figures 6 and 7. In the experiments on CIFAR-10, after 40 epochs of fine tuning, our method maintained the lowest P-value. Additionally, both our method and UBW achieved a WSR of over 90%. In the experiments on Tiny-ImageNet and PubFig, only the proposed method remained effective. After 100 epochs of fine tuning, our WSR stayed above 85%, and the p-value consistently stayed below the threshold of 0.05, allowing the watermark to be effectively verified. Therefore, the proposed method has good robustness against fine tuning attacks.

The comparison of robustness against pruning attacks is shown in Figures 8 and 9. In the experiments on CIFAR-10, when the pruning rate is below 80%, our method has the lowest p-value and the highest WSR. The proposed watermarking process is effective until a pruning rate of 90%. In the experiments on Tiny-ImageNet, within a pruning rate of 28%, our p-value remains below 1.47×10^{-24} and the WSR remains above 76%, where the watermark still can be verified. In the experiments on PubFig, our p-value remained below 6.49×10^{-34} until pruning reached 84%, and the WSR stayed close to 100% until pruning reached 91%. Same as above, the comparison methods are ineffective on Tiny-ImageNet and PubFig. Therefore, the proposed method has better robustness against pruning attacks.



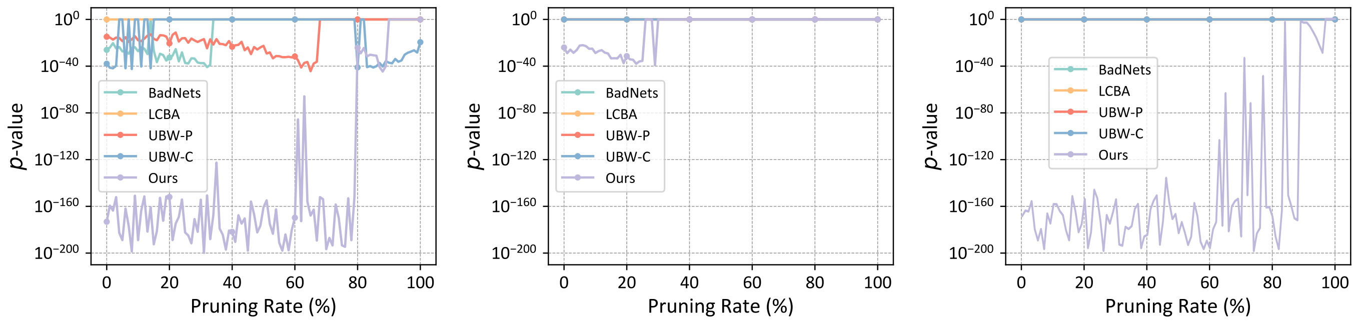
(**a**) CIFAR-10 (**b**) Tiny ImageNet (**c**) PubFig

Figure 6. Comparison of p-value in robustness against fine tuning attacks.



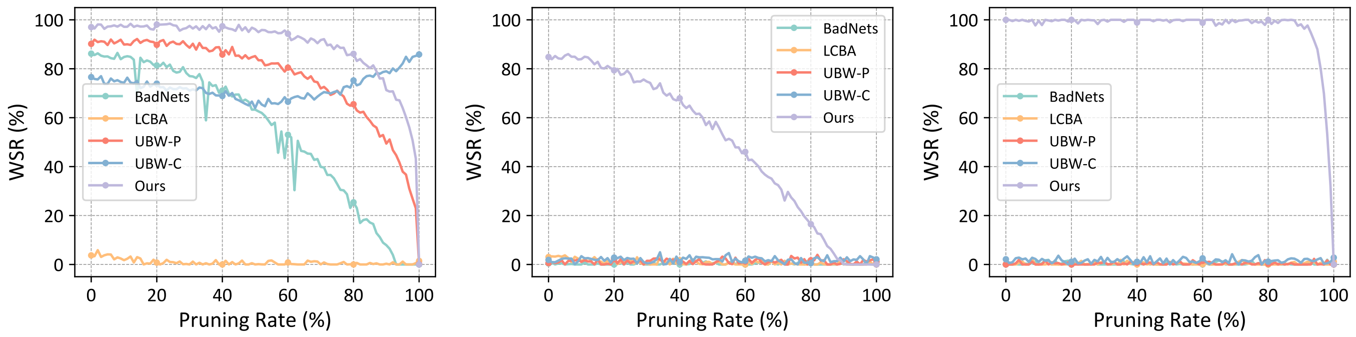
(**a**) CIFAR-10 (**b**) Tiny ImageNet (**c**) PubFig

Figure 7. Comparison of WSR in robustness against fine tuning attacks.



(**a**) CIFAR-10 (**b**) Tiny ImageNet (**c**) PubFig

Figure 8. Comparison of $p-$value in robustness against pruning attacks.



(**a**) CIFAR-10 (**b**) Tiny ImageNet (**c**) PubFig

Figure 9. Comparison of WSR in robustness against against pruning attacks.

4.4. Ablation Studies

4.4.1. Trigger Budgets

The amplitude of watermark triggers can affect the model's ability to learn trigger features. Theoretically, the higher the amplitude of the trigger, the higher the WSR; however, this leads to lower imperceptibility. To find an appropriate ϵ of ℓ^∞-norm, we conducted experiments with different trigger amplitudes using CIFAR-10. The results are shown in Figure 10.

It indicates that the WSR increases as the trigger budget increases, while the MA and TA remain largely unchanged. When the ℓ^∞-norm reached 16/255, the WSR reached 100%. As the ℓ^∞-norm increased further, the WSR remained stable at 100%. Therefore, in our study, we set ϵ of ℓ^∞-norm to 16/255.

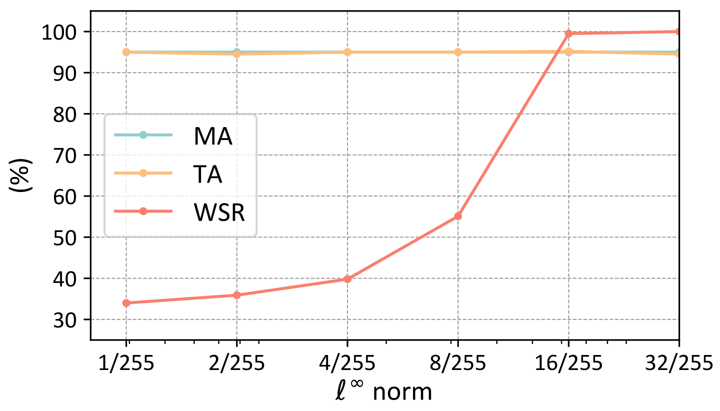


Figure 10. Ablation study on trigger budgets in terms of ℓ^∞-norm.

4.4.2. Misalignment of Surrogate and Target Models

In this section, we analyze the performance of our method when the structure of the surrogate model differs from that of the target model. We do this to find out if the surrogate model using a different model structure compared to the target model would reduce the performance of the proposed dataset-watermarking method. We conducted experiments using commonly used classification models: ResNet-18 [31], VGG-16 [34], and MobileNet V3 Small [35]. We conducted experiments separately for each of the three models as either the surrogate model or the target model. A total of nine experiments were carried out using CIFAR-10. The results are shown in Figure 11.

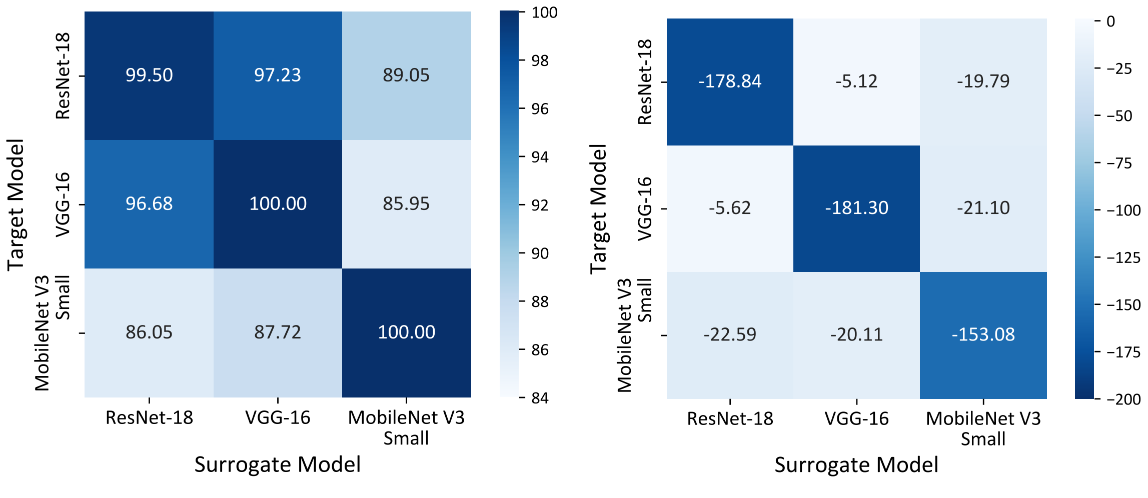


Figure 11. WSR and $p-$value results between different surrogate models and target models.

We observed that even when the structure of the surrogate model differs from the target model, our method still maintains a WSR greater than 85% and all p-values are less than 0.05. Although it achieves the best performance when the structure of the surrogate model is the same as the target model, the watermark can be successfully verified in all experiments. This suggests that the defender does not need to be concerned with the specific model used by the adversary.

4.5. Analysis of Limitations

Although the proposed method demonstrates superior effectiveness, imperceptibility, and robustness compared with existing methods, it still has limitations.

When there is a significant difference between the public OOD dataset used for training the surrogate model and the target dataset, the effectiveness of the watermark may decrease.

This reliance on the quality and relevance of the public OOD dataset may, to some extent, limit the applicability of the watermark in real-world scenarios.

Addressing this limitation is essential for enhancing the robustness and applicability of dataset watermarking in broader adversarial scenarios, which are noted down to be explored in future research.

5. Conclusions

This study proposes a clean-label dataset-watermarking method based on trigger optimization. We perform iterative optimization of the trigger using a surrogate model, guided by target class samples. The optimized triggers carry the robust feature representations of the target class. This optimization of triggers significantly enhances the algorithm's imperceptibility and robustness, ensuring its effectiveness even at very low watermarking rates. At a watermarking rate of 0.05%, with only 25, 50, and 6 samples embedded with triggers in CIFAR-10, Tiny-ImageNet, and PubFig, respectively, the proposed method ensures effectiveness and high robustness. This significantly enhances the practicality of the dataset-watermarking method. Extensive experiments on CIFAR-10, Tiny-ImageNet, and PubFig datasets proved that, compared to existing methods, the proposed method exhibits the best performance in terms of imperceptibility and robustness, and has a low watermarking rate. Our study provides a reliable tool for protecting the intellectual property of datasets in machine learning.

Author Contributions: Conceptualization, W.C. and Y.X.; methodology, W.C. and G.W.; software, G.W.; validation, G.W. and X.X.; formal analysis, H.P.; investigation, Y.S.; writing—original draft preparation, W.C. and G.W.; writing—review and editing, X.X. and Y.X.; supervision, W.C.; funding acquisition, W.C., Y.X. and H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant Nos. 42201444 and 42271431), the Hunan Provincial Natural Science Foundation of China (Grant No. 2024JJ8334), and the open fund of State Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University (Grant No. 22S01).

Data Availability Statement: The data associated with this research are available online. The CIFAR-10 dataset is available at https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 8 May 2024). The Tiny-ImageNet dataset is available at https://cs231n.stanford.edu/ (accessed on 8 May 2024). The PubFig dataset is available at https://www.cs.columbia.edu/CAVE/databases/pubfig/ (accessed on 8 May 2024). At the same time, we have made appropriate citations in this paper.

Acknowledgments: The authors would like to thank the anonymous reviewers for their comments and helpful suggestions.

Conflicts of Interest: Author Haibo Peng and author Yingchen She are employed by the Third Surveying and Mapping Institute of Hunan Province. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Wang, J.; Ma, Y.; Zhang, L.; Gao, R.X.; Wu, D. Deep learning for smart manufacturing: Methods and applications. *J. Manuf. Syst.* **2018**, *48*, 144–156. [CrossRef]
2. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. In Proceedings of the International Conference on Learning Representations, Virtual, 3–7 May 2021.
3. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR , Long Beach, CA, USA, 10–15 June 2019; pp. 6105–6114.
4. Janiesch, C.; Zschech, P.; Heinrich, K. Machine learning and deep learning. *Electron. Mark.* **2021**, *31*, 685–695. [CrossRef]
5. Liang, W.; Tadesse, G.A.; Ho, D.; Fei-Fei, L.; Zaharia, M.; Zhang, C.; Zou, J. Advances, challenges and opportunities in creating data for trustworthy AI. *Nat. Mach. Intell.* **2022**, *4*, 669–677. [CrossRef]
6. Roh, Y.; Heo, G.; Whang, S.E. A survey on data collection for machine learning: A big data-ai integration perspective. *IEEE Trans. Knowl. Data Eng.* **2019**, *33*, 1328–1347. [CrossRef]

7. Abdollahi, A.; Pradhan, B.; Shukla, N.; Chakraborty, S.; Alamri, A. Deep learning approaches applied to remote sensing datasets for road extraction: A state-of-the-art review. *Remote Sens.* **2020**, *12*, 1444. [CrossRef]
8. Maini, P.; Yaghini, M.; Papernot, N. Dataset inference: Ownership resolution in machine learning. *arXiv* **2021**, arXiv:2104.10706.
9. Ali, A.; Pinciroli, R.; Yan, F.; Smirni, E. Batch: Machine learning inference serving on serverless platforms with adaptive batching. In Proceedings of the SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Virtual, 9–19 November 2020; IEEE: Piscataway, NJ, USA , 2020; pp. 1–15.
10. Wu, C.J.; Brooks, D.; Chen, K.; Chen, D.; Choudhury, S.; Dukhan, M.; Hazelwood, K.; Isaac, E.; Jia, Y.; Jia, B.; et al. Machine learning at facebook: Understanding inference at the edge. In Proceedings of the 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), Washington, DC, USA, 16–20 February 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 331–344.
11. Taheri, R.; Shojafar, M.; Arabikhan, F.; Gegov, A. Unveiling vulnerabilities in deep learning-based malware detection: Differential privacy driven adversarial attacks. *Comput. Secur.* **2024**, *146*, 104035. [CrossRef]
12. Begum, M.; Uddin, M.S. Digital image watermarking techniques: A review. *Information* **2020**, *11*, 110. [CrossRef]
13. Deng, H.; Qin, Z.; Wu, Q.; Guan, Z.; Deng, R.H.; Wang, Y.; Zhou, Y. Identity-based encryption transformation for flexible sharing of encrypted data in public cloud. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3168–3180. [CrossRef]
14. Li, Y.; Zhu, M.; Yang, X.; Jiang, Y.; Wei, T.; Xia, S.T. Black-box dataset ownership verification via backdoor watermarking. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2318–2332. [CrossRef]
15. Chen, X.; Liu, C.; Li, B.; Lu, K.; Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv* **2017**, arXiv:1712.05526.
16. Gu, T.; Liu, K.; Dolan-Gavitt, B.; Garg, S. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* **2019**, *7*, 47230–47244. [CrossRef]
17. Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; Shmatikov, V. How to backdoor federated learning. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Virtual, 26–28 August 2020; pp. 2938–2948.
18. Liu, Y.; Ma, X.; Bailey, J.; Lu, F. Reflection backdoor: A natural backdoor attack on deep neural networks. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020 ; Springer: Berlin/Heidelberg, Germany, 2020; pp. 182–199.
19. Li, S.; Xue, M.; Zhao, B.Z.H.; Zhu, H.; Zhang, X. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Trans. Dependable Secur. Comput.* **2020**, *18*, 2088–2105. [CrossRef]
20. Li, Y.; Zhang, Z.; Bai, J.; Wu, B.; Jiang, Y.; Xia, S.T. Open-sourced dataset protection via backdoor watermarking. *arXiv* **2020**, arXiv:2010.05821.
21. Zeng, Y.; Pan, M.; Just, H.A.; Lyu, L.; Qiu, M.; Jia, R. Narcissus: A practical clean-label backdoor attack with limited information. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, Copenhagen, Denmark, 26–30 November 2023; pp. 771–785.
22. Tang, R.; Feng, Q.; Liu, N.; Yang, F.; Hu, X. Did you train on my dataset? towards public dataset protection with cleanlabel backdoor watermarking. *Acm Sigkdd Explor. Newsl.* **2023**, *25*, 43–53. [CrossRef]
23. Turner, A.; Tsipras, D.; Madry, A. Label-consistent backdoor attacks. *arXiv* **2019**, arXiv:1912.02771.
24. Souri, H.; Fowl, L.; Chellappa, R.; Goldblum, M.; Goldstein, T. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 19165–19178.
25. Taheri, R.; Javidan, R.; Shojafar, M.; Pooranian, Z.; Miri, A.; Conti, M. On defending against label flipping attacks on malware detection systems. *Neural Comput. Appl.* **2020**, *32*, 14781–14800. [CrossRef]
26. Li, Y.; Bai, Y.; Jiang, Y.; Yang, Y.; Xia, S.T.; Li, B. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 13238–13250.
27. Krizhevsky, A.; Hinton, G. Learning Multiple Layers of Features from Tiny Images. Master's Thesis, University of Toronto, Toronto, ON, Canada , 2009.
28. Le, Y.; Yang, X. Tiny imagenet visual recognition challenge. *CS 231N* **2015**, *7*, 3.
29. Kumar, N.; Berg, A.C.; Belhumeur, P.N.; Nayar, S.K. Attribute and simile classifiers for face verification. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 365–372.
30. Liu, Z.; Luo, P.; Wang, X.; Tang, X. Deep learning face attributes in the wild. In Proceedings of the IEEE International Conference on Computer Vision, ICCV, Santiago, Chile, 7–13 December 2015; pp. 3730–3738.
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
32. Saha, A.; Subramanya, A.; Pirsiavash, H. Hidden trigger backdoor attacks. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 11957–11965.
33. Zhang, R.; Isola, P.; Efros, A.A.; Shechtman, E.; Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 586–595.

34. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
35. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324.