electronics

MDPI

Article

Feature-Decoupled Contrastive Self-Distillation for Black-Box Watermark Removal

Wei Wang [1], Shaofei Yang [1,*], Jiawei Zhu [1], Siyuan Zhu [2], Yang Bai [2] and Jinyin Chen [2]

[1] National Key Laboratory of Electromagnetic Space Security, Jiaxing 314000, China; wwzwh@163.com (W.W.); z_jw_1212@163.com (J.Z.)
[2] College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China; 302023103015@zjut.edu.cn (S.Z.); 211124030094@zjut.edu.cn (Y.B.); chenjinyin@zjut.edu.cn (J.C.)
* Correspondence: w_w_101@163.com

Abstract

Deep neural networks have been widely deployed in numerous real-world applications, making trained models valuable intellectual property. However, embedded watermarks remain vulnerable to removal attacks, particularly when an adversary has only limited clean unlabeled data and the original watermark information is unavailable. To address this problem, this paper proposes a feature-decoupled contrastive self-distillation method for watermark removal against black-box-verifiable watermarks under an editable-model threat model. Given clean unlabeled data, high-confidence pseudo-labels are first generated solely from the output predictions of the watermarked model, without requiring ground-truth labels. Feature decoupling is then employed to distinguish normal and watermark-related features, based on which an inferred watermark-related perturbation is obtained without assuming exact recovery of the original secret trigger. Contrastive learning subsequently guides the editable model away from watermark-related representations, while feature- and label-level self-distillation provide layer-wise regularization across intermediate representations and predictions to preserve the model's normal task behavior. Experiments on four datasets (MNIST, CIFAR-10, ImageNet-10, and ImageNet-100) demonstrate that the proposed method effectively reduces residual watermark behavior under the limited-data, unlabeled setting. For example, the verification success rate of the BadNets watermark is reduced to 4.0% and 5.67% on CIFAR-10 and ImageNet-100, respectively. For the complete method, the largest reported mean main-task accuracy drop relative to the pre-removal watermarked model is 3.95 percentage points in the main comparison and 4.98 percentage points when the data-proportion sensitivity experiments are included. Visualization and ablation results further demonstrate the effectiveness of the feature-decoupling and self-distillation components for watermark removal.

Keywords: watermark attack; contrastive learning; self-distillation; black-box watermarking; trigger inversion; feature decoupling

Check for updates

Academic Editor: Juan-Carlos Cano

Received: 20 September 2026
Revised: 24 October 2026
Accepted: 29 October 2026
Published: 30 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Deep neural networks (DNNs) have been widely applied in many fields, such as computer vision, speech processing, natural language processing, and autonomous driving, and have become an indispensable part of modern intelligent systems. However, training a large-scale DNN model incurs expensive computational and engineering costs, which makes the trained model an important asset of AI companies. As a result, DNN models face the risk of model stealing attacks [1,2], and the demand for model copyright protection is extremely urgent.

Therefore, researchers have proposed many DNN watermarking methods [3] to trace stolen model copies. In general, a model watermarking scheme consists of two parts: watermark embedding and watermark verification. In the embedding stage, secret identity information, i.e., the watermark, is embedded into the target model as training proceeds. In the verification stage, the watermark is verified, and ownership is determined based on whether the same or similar watermark is detected in the suspicious model.

Depending on the information required in the verification stage, watermarking techniques can be mainly divided into two categories: white-box watermarking and black-box watermarking. White-box watermarking requires access to the internal parameters and structural details of the model during verification, and extracts the watermark by analyzing the specific parameters of the model. However, in practical applications, deep learning models are usually deployed in a black-box manner, providing services only through remote APIs without disclosing internal details of the model, which greatly limits the application of white-box watermarking in real-world scenarios. In contrast, black-box watermarking relies solely on the outputs of the model for verification and does not require access to its internal information; it is therefore more in line with practical deployment requirements and has gradually become the mainstream direction of watermarking research.

However, black-box watermarking techniques also face severe security challenges. Attackers can attack watermarked models through various means, attempting to destroy or remove the watermark information and thereby bypass copyright verification. Therefore, studying the mechanisms and methods of black-box watermark attacks is of great significance for designing more robust watermark embedding schemes.

To realize an attack against black-box watermarking, the following constraints usually need to be satisfied: (1) the attack process should not consume more computational resources and time than training a DNN model from scratch; (2) the attacked model should not suffer an obvious degradation in main-task performance, so as to ensure that the functionality of the model is not affected. These constraints drive attackers to adopt efficient and stealthy attack means, such as model fine-tuning, pruning, and distillation, so as to destroy or remove the embedded watermark information while preserving the model performance as much as possible. Such attacks not only pose a severe challenge to model copyright protection, but also motivate researchers to further explore more robust watermark embedding methods to cope with increasingly complex attack scenarios.

At present, the main black-box watermark removal methods include fine-pruning [4] and neural attention distillation [5]. Among them, fine-pruning integrates pruning and fine-tuning, which can effectively eliminate the watermark trigger patterns in the model. It first prunes the neurons associated with the trigger from the watermarked model to purify the model, and then fine-tunes the model with a small amount of clean data to restore its performance on the original samples. Neural attention distillation removes the watermark trigger through knowledge distillation with the help of a small amount of clean data. Since knowledge distillation involves two models (i.e., the teacher model and the student model), backdoor defense based on knowledge distillation requires selecting appropriate teacher and student models, and the core of this method lies in finding a suitable attention representation to ensure the effectiveness of the distillation defense. However, existing watermark removal pipelines cannot thoroughly remove the model watermark from shallow to deep layers, and instead further degrade the performance of the model. In addition, most existing methods remove backdoor watermarks in scenarios with abundant clean data and relatively complete labels, and cannot achieve effective watermark removal when only a small amount of clean data is available and labels are missing.

Based on the above problems, this paper proposes a self-distillation watermark attack method based on contrastive learning. First, conventional data augmentation operations

such as rotation, translation, and scaling are applied to expand the dataset, and pseudo-labels are obtained through confidence-based filtering. Then, based on feature decoupling, the method inverts the watermark trigger, and feeds the original data and the watermarked data into the target network—the object of the watermark attack—together with a replica network with frozen parameters. The positive pair consists of the output features of the target network on watermarked inputs and those of the replica network on clean samples, while the negative pair consists of the output features of the target network on watermarked inputs and those of the replica network on watermarked inputs. By minimizing the features of the positive pair and maximizing the features of the negative pair, the watermark attack is achieved. Then, a fully connected layer is added to each sub-module of the model, with its output dimension consistent with that of the deepest layer of the model, so that the output feature map of each sub-module becomes closer to that of the deepest layer, thereby realizing feature distillation. Next, a small amount of pseudo-labeled data is used to fine-tune the model, and a fully connected network is appended after each sub-module to output the classification confidence of each class, making these confidences similar to those of the final model output, thereby realizing label self-distillation. Through the above steps, watermark removal is achieved in the scenario of missing labels while the main-task performance of the model is guaranteed.

The main contributions of this paper are summarized as follows:

- A contrastive-learning-based self-distillation watermark attack method is proposed, wherein high-confidence pseudo-labels are generated through data augmentation and confidence-based filtering, thereby enabling effective watermark removal in scenarios with only limited clean data and missing labels.
- A feature-decoupling-based trigger inversion method is designed to separate normal features from watermark features in a watermarked model and to invert the watermark trigger without access to the original watermark information.
- Feature self-distillation and label self-distillation are introduced to jointly remove the watermark from shallow to deep layers at both the feature and label levels, while preserving the model's main-task performance.
- Extensive experiments across four datasets (MNIST, CIFAR-10, ImageNet-10, and ImageNet-100) and four black-box watermarking methods (BadNets, Blend, CTW, and WaNet) demonstrate that the proposed method reduces the verification success rate of the BadNets watermark to 4.0% and 5.67% on CIFAR-10 and ImageNet-100, respectively. For the complete method, the largest reported mean main-task accuracy drop is 3.95 percentage points in the main comparison and 4.98 percentage points when the data-proportion sensitivity experiments are included.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the preliminaries. Section 4 presents the proposed methodology. Section 5 reports the experimental setup and results. Section 6 concludes this paper.

2. Related Work

2.1. DNN Watermarking

Deep neural networks have become valuable intellectual property, and watermarking has been widely studied as a means of protecting their ownership [3]. According to the information required during verification, existing watermarking schemes are usually divided into white-box and black-box approaches. White-box watermarking embeds the watermark into the internal parameters or structural characteristics of the model, and extracts it by analyzing the model weights [6–8]. Although such schemes can carry a relatively large

amount of watermark information, they require access to the internal details of the model, which is often unavailable for models that are deployed as remote services.

Black-box watermarking instead verifies ownership solely through the input-output behavior of the model and is therefore more consistent with practical deployment scenarios, and recent studies further extend model ownership protection to fingerprinting and watermarking of deployed services [9,10]. A representative family of black-box methods embeds a trigger set into the model during training, so that the model behaves abnormally on inputs carrying a predefined trigger while maintaining normal performance on clean inputs. BadNets [11] and Blend [12] construct the trigger set by stamping a patch or blending a pattern into benign samples, and WaNet [13] adopts an imperceptible warping-based trigger. CTW [14] employs a parameterized transformation to embed a robust and verifiable watermark into a neural network, and has been widely used as a benchmark for evaluating watermarking and watermark-removal methods. These trigger-set-based watermarks are effective and easy to verify [15,16], but they also expose an attack surface: since the watermark is encoded in the model's response to a specific trigger, it may be removed or destroyed by an adversary who is able to modify the model, and a variety of backdoor and trojan attacks have demonstrated the vulnerability of DNN models to such manipulations [17–19].

2.2. Watermark Removal and Backdoor Defense

Watermark removal attacks aim to eliminate the watermark from a watermarked model while preserving its main-task performance. Fine-tuning (FT) [20] mixes a small number of watermarked samples with correct labels into the clean data and continues training, which weakens the association between the trigger and the target label. Fine-Pruning (FP) [4] first prunes the neurons that are dormant on clean inputs and then fine-tunes the pruned model. Neural Cleanse (NC) [21] detects and filters abnormal inputs by estimating the minimal perturbation required to change the model prediction. Neural Attention Distillation (NAD) [5] uses a fine-tuned teacher network to guide the fine-tuning of the watermarked student network on a small set of clean data, so that the attention of the intermediate layers of the student network is aligned with that of the teacher network. Beyond these methods, a series of backdoor defenses and trigger inversion techniques have been developed from the perspectives of input filtering, model diagnosis, and training-set purification [22–25].

Despite their effectiveness, existing watermark removal methods [26–31] still suffer from two limitations. First, they mainly suppress the watermark at specific layers or at the final prediction level, and cannot thoroughly remove the watermark from shallow to deep layers, so that residual watermark features may still remain in the intermediate representations. Second, most of them assume abundant clean data with relatively complete labels, and their performance degrades sharply when only a small amount of clean data is available and labels are missing.

2.3. Contrastive Learning and Knowledge Distillation

Contrastive learning learns discriminative feature representations by pulling similar samples closer and pushing dissimilar samples apart, and has achieved remarkable success in self-supervised settings where labels are scarce [32,33]. A representative framework, SimCLR [32], treats different augmentations of the same image as a positive pair and the remaining samples in the batch as negative pairs, and optimizes a normalized temperature-scaled cross-entropy loss. Since contrastive learning can learn high-quality representations without labels, it provides a natural tool for separating watermark-related features from normal features in unlabeled scenarios.

Knowledge distillation transfers the knowledge of a large teacher model to a compact student model by matching their predictions [34]. It has been widely used for model compression and, more recently, for backdoor defense, where the behavior of the teacher on clean data guides the student away from trigger-related behavior. Self-distillation further removes the need for an external teacher by aligning the outputs of the intermediate layers or sub-modules with those of the deepest layer of the same network [35–38], which encourages every layer to learn representations that are consistent with the final prediction.

Different from the above methods, the proposed method combines contrastive learning with feature self-distillation and label self-distillation, so that the watermark is removed at both the feature level and the prediction level from shallow to deep layers, and the attack remains effective in the scenario where only a small amount of clean data is available and labels are missing.

3. Preliminary

3.1. Formulation

Watermark Attack: A watermark attack refers to modifying or perturbing the weights, structure, or outputs of a model so that the watermark information originally embedded in it is destroyed or can no longer be effectively identified. Its purpose is usually to remove privacy information such as copyright protection and identity authentication from the model. In deep learning models, watermark embedding usually involves adding certain specific identification information during training, so that this information can be extracted later through some detection methods. A watermark attack makes this identification information disappear by changing the structure or parameters of the model. It can be regarded as an optimization process, and its formulation is as follows:

$$\min_{\theta'}[L_{\text{task}}(\theta', D) + \lambda \cdot L_{\text{wm}}(\theta', W)] \tag{1}$$

where θ' denotes the modified model parameters; L_{task} denotes the loss function on the original task (e.g., classification or regression), which is used to preserve model performance; D denotes the original training dataset; L_{wm} denotes the watermark attack function, which is used to eliminate the watermark features of the model; W denotes the constraints related to the watermark trigger set or watermark detector (e.g., parameter distribution or specific input-output responses); and λ is a hyperparameter that controls the strength of the watermark attack and the preservation of model performance.

Contrastive Learning: Contrastive learning is a self-supervised learning method that learns feature representations by learning to distinguish the similarities and differences between different samples or samples of different classes. Its core idea is to optimize the network so that the feature representations of similar samples become closer, while those of dissimilar samples are pushed as far apart as possible. Contrastive learning is commonly used in unsupervised learning tasks; in particular, it can learn high-quality feature representations in scenarios with missing labels. Its core concepts include positive/negative sample pairs and similarity measures. A positive pair in contrastive learning refers to samples with similar features or semantics, e.g., different augmentations (such as rotation, scaling, and color changes) of the same image, or samples belonging to the same class. A negative pair refers to samples with large differences in features or semantics, and the goal is to push these samples as far apart as possible in the feature space through contrastive learning. Contrastive learning measures the similarity between samples through a similarity measure (usually Euclidean distance or cosine similarity). In the classical SimCLR framework, the contrastive loss is defined as:

$$L_{\text{SimCLR}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{\text{sim}(z_i, z_j)/\tau}}{\sum_{k=1}^{2N} \mathbf{1}_{k \neq i} e^{\text{sim}(z_i, z_k)/\tau}} \tag{2}$$

where N is the number of samples in each batch; z_i and z_j form a positive pair; and $\mathrm{sim}(\cdot)$ denotes the similarity measure function.

Knowledge Distillation: Model distillation is a model compression technique in deep learning that aims to transfer the knowledge of a complex and large "teacher model" to a smaller and more computationally efficient "student model", so that the student model can approach the teacher model in performance while significantly reducing computational and storage requirements, thereby enabling efficient inference and deployment. In this way, the student model inherits the knowledge of the teacher model, especially its complex features and decision patterns. Its loss function usually consists of two parts: the difference between the predictions of the student model and the ground-truth labels, and the difference between the predictions of the student model and those of the teacher model. The loss between the student model and the ground-truth labels is formulated as follows:

$$\mathrm{L}_{\mathrm{hard}} = \mathrm{E}_{(x,y)\sim \mathrm{D}}[\mathrm{CE}(f_S(x), y)] \tag{3}$$

where $f_S(x)$ is the output of the student model; y is the ground-truth label; CE is the cross-entropy loss function; and $\mathcal{D}$ is the training dataset. The loss between the outputs of the student model and the teacher model is as follows:

$$\mathrm{L}_{\mathrm{soft}} = \mathrm{E}_{x\sim \mathrm{D}}[\mathrm{KL}(f_T(x), f_S(x))] \tag{4}$$

where $f_T(x)$ is the output of the teacher model; the KL divergence is used to measure the difference between the probability distributions of the teacher model and the student model.

3.2. Threat Model

We consider the removal of a black-box-verifiable watermark from an editable copy of a watermarked model. Here, the term black-box characterizes the watermark verification mechanism: ownership can be verified from the model's input-output behavior without inspecting its internal parameters. It does not imply that the adversary is restricted to output-only queries. The adversary is assumed to possess an editable copy of the watermarked model and can therefore access its architecture, parameters, and intermediate representations. The threat model is defined in terms of the adversary's knowledge, objective, capabilities, and attack scenario as follows.

Attack Knowledge: The adversary knows the task specification and has access to an editable copy of the watermarked model, including its architecture and learned parameters. The adversary can therefore inspect the computational graph and intermediate representations required by the proposed attack. However, the adversary does not know the original watermark trigger, the watermark-generation procedure, or other secret information used to construct the watermark. In particular, the attack does not rely on prior knowledge of the original trigger pattern. The adversary also has no access to the original training process or watermark-embedding procedure beyond the information available from the obtained model and the accessible clean data.

Attack Goal: The primary objective is to remove or substantially weaken the embedded watermark while preserving the utility of the protected model. Specifically, the adversary aims to reduce the watermark verification success rate while maintaining the model's performance on its original task. The attack is therefore formulated as a utility-preserving watermark removal problem rather than a general model degradation or model-replacement attack.

Attack Capabilities: We consider a practical scenario in which an adversary obtains an editable copy of a watermarked model but does not possess the secret information used to construct or verify its watermark. The model can be modified locally by the adversary, while the watermark remains unknown. The attack begins by using the model's outputs

on clean unlabeled data to construct high-confidence pseudo-labels. The adversary then exploits the internal representations of the editable model to perform trigger inversion, feature decoupling, contrastive optimization, and self-distillation, ultimately producing a modified model with reduced watermark effectiveness and preserved task performance. Under this setting, black-box refers exclusively to watermark verification, whereas the attack assumes editable model access.

4. Methodology

4.1. Overview

This paper proposes a self-distillation watermark attack method based on contrastive learning. The core of this method is based on the idea of contrastive learning: during model updating, the features of watermarked samples are pushed away from the initial watermark features and closer to the features of normal samples, thereby realizing the watermark attack in an unlabeled scenario. In addition, the method introduces self-distillation to achieve all-round feature and label self-distillation from shallow to deep layers, so as to thoroughly remove the watermark while preserving the main-task performance. The system diagram is shown in Figure 1.

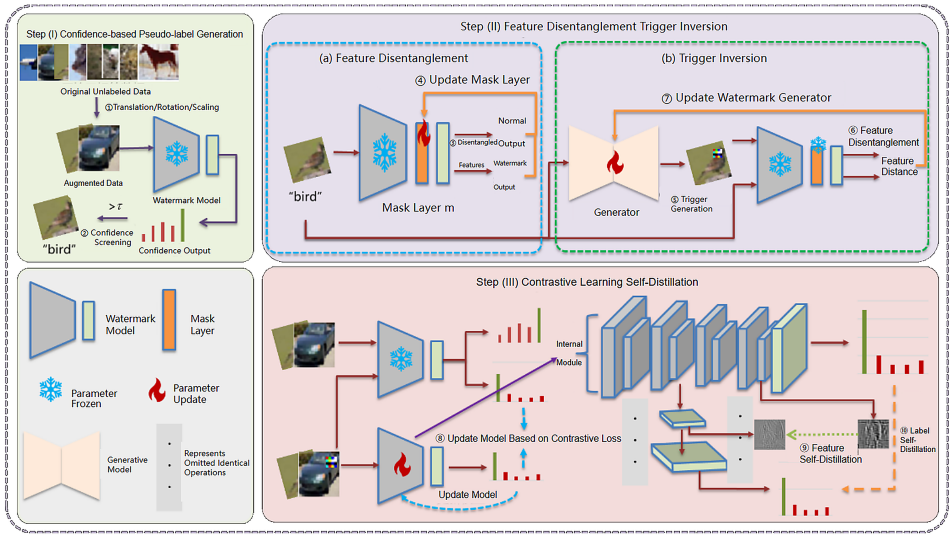


Figure 1. Framework of the self-distillation watermark attack method based on contrastive learning.

4.2. Pseudo-Label Generation Based on Confidence-Based Filtering

As illustrated in Figure 1, Step (I) takes as input a batch of unlabeled normal data $D = \{x_1, x_2, \ldots, x_N\}$ and a watermarked model f, where x_i denotes the ith sample. Each unlabeled sample x_i is augmented by geometric transformations (translation, rotation, and scaling) to generate augmented data, yielding $\hat{D} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N\}$. The model f is then used to predict the output probability distribution $p_i = f(\hat{x}_i)$, from which the predicted label $\hat{y}_i$ and the confidence score s_i are extracted:

$$\begin{cases} \hat{y}_i = \arg\max(p_i) \\ s_i = \max(p_i) \end{cases} \tag{5}$$

A threshold τ is then given to filter high-confidence samples. For each sample $\hat{x}_i$, check whether its confidence score s_i satisfies $s_i > \tau$; if so, $(\hat{x}_i, \hat{y}_i)$ is added to the high-confidence pseudo-label dataset $D_{high} = \{(\hat{x}_j, \hat{y}_j)\}$, thereby completing the pseudo-label generation based on confidence-based filtering.

The pseudo-label generation stage only requires access to the final outputs of the watermarked model. Given the available clean unlabeled data, the adversary obtains the

model prediction probability distribution and uses the predicted class and its confidence score to construct the high-confidence pseudo-labeled dataset $\mathcal{D}_{\text{high}}$. No intermediate representations or parameter modifications are required at this stage. The resulting dataset contains only clean samples and their corresponding model-generated pseudo-labels, which do not require access to ground-truth labels. These high-confidence pseudo-labeled samples are then used in the subsequent feature-level attack stages, where editable access to the obtained model is required. Thus, all optimization stages rely on $\mathcal{D}_{\text{high}}$ rather than manually labeled data.

4.3. Trigger Inversion Based on Feature Decoupling

As depicted in module (a) of Step (II) in Figure 1, a mask layer m is introduced before the fully connected layer of the editable copy of the watermarked model. Subsequently, with the high-confidence pseudo-labeled samples generated in Section 4.2, the normal and watermark-related features of the watermarked model are disentangled. Specifically, the mask is optimized such that the retained features preserve the normal classification capability, while the complementary features contribute less to the normal classification task. To avoid a degenerate solution in which the mask approaches an all-one matrix, we additionally impose sparsity and size constraints on the mask. The formulation is given as follows:

$$\begin{aligned} \underset{m}{\operatorname{argmin}} \quad & \sum_{(x,y)\in\mathcal{D}_{\text{high}}} [\mathrm{L}(F_b \circ (F_a(x) \odot m), y) - \mathrm{L}(F_b \circ (F_a(x) \odot (1-m)), y)] \\ & + \lambda_m \|m\|_1 \end{aligned} \tag{6}$$

This is subject to

$$0 \le m \le 1, \qquad \frac{\|m\|_1}{|m|} \le \rho, \tag{7}$$

where $F_a(x)$ denotes the features of the input sample extracted by the convolutional layers, F_b denotes the classification mapping of the fully connected layer over the features, and m denotes a mask matrix with the same dimension as the features. The elements of m lie in $[0, 1]$, where a larger value indicates that the corresponding feature is retained. λ_m controls the sparsity regularization, and ρ specifies the maximum average activation allowed for the mask. By minimizing Equation (6), the features covered by the mask m, i.e., $F_a(x) \odot m$, are encouraged to preserve the normal classification capability, whereas the complementary features, i.e., $F_a(x) \odot (1-m)$, are encouraged to contribute less to the normal classification task. The parameters of F_a and F_b are kept fixed during this optimization, such that only the auxiliary mask m is updated.

Subsequently, based on the obtained feature decomposition, watermarked samples are generated such that they remain similar to the original samples in the retained features while inducing a larger change in the complementary features. Let $\mathcal{D}_l = \mathcal{D}_{\text{high}}$ denote the clean samples together with their corresponding high-confidence pseudo-labels generated in Section 4.2. The trigger generator G_θ is optimized as follows:

$$\begin{cases} \min\limits_{\theta} \sum\limits_{(x,y)\in\mathcal{D}_l} \Big(\|(F_a(x) - F_a(G_\theta(x))) \odot m\| \\ \qquad\qquad - \|(F_a(x) - F_a(G_\theta(x))) \odot (1-m)\| \Big) \\ \text{s.t.} \quad \|x - G_\theta(x)\| \le \epsilon_{\text{inv}}. \end{cases} \tag{8}$$

where G_θ denotes the trigger generator, and ϵ_{inv} denotes the maximum allowed distance between the generated sample and the original sample in the input space during trigger inversion. The confidence threshold used for pseudo-label filtering in Section 4.2 is denoted by τ_{conf} and is not reused as the input-distance bound in Equation (8). During the optimization of Equation (8), the parameters of the watermarked model and the learned mask m are kept fixed, while gradients are propagated through the differentiable feature extractor F_a to the trigger generator G_θ. Thus, only the parameters θ of G_θ are updated in this stage.

By optimizing Equation (8), the generated samples are encouraged to preserve the retained feature representation while producing larger changes in the complementary feature representation. The resulting perturbation is defined as

$$\delta_\theta(x) = G_\theta(x) - x. \tag{9}$$

We refer to $\delta_\theta(x)$ as an inferred watermark-related perturbation. Importantly, the inferred perturbation is not assumed to be an exact reconstruction of the original secret watermark trigger, since the trigger-inversion objective only seeks a model-derived perturbation that exposes watermark-related behavior. Therefore, the purpose of trigger inversion is to obtain an effective perturbation for constructing triggered samples in the subsequent contrastive learning and self-distillation procedures, rather than to recover the exact trigger used during watermark embedding.

4.4. Feature and Label Self-Distillation

As depicted in modules (b) and (c) of Step (II) in Figure 1, feature-level and label-level self-distillation are introduced to further guide the watermark-removal process across different layers of the editable model. Given the inferred watermark-related perturbation obtained in Section 4.3, triggered samples are constructed from the clean samples in $\mathcal{D}_l$. To avoid using the outputs of a simultaneously updating network as moving targets, we maintain two copies of the model: a frozen reference model f_{ref} and a trainable target model f_{stu}. The reference model is initialized from the editable watermarked model before the watermark-removal optimization and remains fixed throughout the training process, whereas the target model is updated by the proposed removal objective.

For a clean sample x_i and its corresponding triggered samples $\{x_{ij}^{tri}\}_{j=1}^{K}$ generated using the inferred perturbation, the frozen reference model produces intermediate representations and output predictions that are used as fixed reference targets. Specifically, let $z_i^{tri,\text{ref}}$ and $y_i^{tri,\text{ref}}$ denote the intermediate feature representation and output prediction of the reference model for the clean sample x_i, respectively, while $z_{ij}^{tri,\text{stu}}$ and $y_{ij}^{tri,\text{stu}}$ denote the corresponding representations and predictions produced by the trainable target model for the j-th triggered sample. The feature-level self-distillation loss is formulated as

$$L_{\text{feat}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\text{MSE}\left(z_{ij}^{tri,\text{stu}}, \text{sg}\left(z_i^{tri,\text{ref}}\right)\right), \tag{10}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operation. Thus, the feature representation of the triggered sample produced by the trainable target model is encouraged to remain consistent with the corresponding fixed reference representation of the clean sample.

Similarly, to preserve the normal prediction behavior while reducing the influence of the inferred watermark-related perturbation, we introduce a label-level self-distillation loss:

$$L_{\text{label}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\text{MSE}\left(y_{ij}^{tri,\text{stu}}, \text{sg}\left(y_i^{tri,\text{ref}}\right)\right). \tag{11}$$

where $y_i^{tri,\text{ref}}$ serves as a fixed reference prediction rather than a watermark-free ground-truth label. The stop-gradient operation ensures that no gradient is propagated through the reference model. Therefore, the self-distillation losses only update the trainable target model and the associated auxiliary layers.

The feature-level and label-level self-distillation losses are jointly optimized with the contrastive learning objective described in the previous section. The overall optimization objective is given by

$$L_{\text{total}} = L_{\text{con}} + \lambda_{\text{feat}}L_{\text{feat}} + \lambda_{\text{label}}L_{\text{label}}, \tag{12}$$

where L_{con} denotes the contrastive learning loss, while λ_{feat} and λ_{label} control the contributions of the feature-level and label-level self-distillation terms, respectively.

During the optimization of Equation (12), the parameters of the reference model f_{ref} are completely frozen. In particular, the intermediate representations $z_i^{tri,\text{ref}}$ and predictions $y_i^{tri,\text{ref}}$ are detached from the computational graph before being used as self-distillation targets. Gradients from L_{feat} and L_{label} are therefore propagated only through the trainable target model f_{stu} and its associated trainable auxiliary layers. The reference model does not receive gradients or parameter updates during the watermark-removal process.

It is important to note that the frozen reference model is not assumed to be watermark-free. Instead, it provides a fixed representation and prediction reference that stabilizes the optimization of the trainable target model. The actual watermark-removal objective is jointly determined by the contrastive loss and the feature- and label-level self-distillation losses. Consequently, the proposed self-distillation mechanism should be understood as a layer-wise regularization strategy that propagates the removal objective across intermediate representations, rather than as a direct proof that the deep reference representations are themselves watermark-free, Algorithm 1 for the specific procedure.

Algorithm 1 Feature-Decoupled Contrastive Self-Distillation for Watermark Removal

Require: Editable watermarked model f_w, unlabeled dataset $\mathcal{D}_u$
Ensure: Watermark-removed model f_{stu}
1: Generate $\mathcal{D}_{\text{high}}$ by pseudo-labeling $\mathcal{D}_u$ with confidence threshold τ_{conf} (Section 4.2)
2: Set $\mathcal{D}_l \leftarrow \mathcal{D}_{\text{high}}$ and optimize the constrained mask m using Equations (6) and (7)
3: Fix m and f_w, and optimize G_θ using Equation (8); obtain $\delta(x) = G_\theta(x) - x$
4: Initialize $f_{\text{stu}} \leftarrow f_w$ and frozen $f_{\text{ref}} \leftarrow f_w$
5: **for** $(x, y) \in \mathcal{D}_l$ **do**
6: Generate triggered samples with G_θ and detach the reference features/predictions from f_{ref}
7: Update f_{stu} by minimizing $L_{\text{total}} = L_{\text{con}} + \lambda_{\text{feat}} L_{\text{feat}} + \lambda_{\text{label}} L_{\text{label}}$
8: **end for**
9: **return** f_{stu}

5. Experiments and Analysis

In this section, we empirically evaluate the proposed method by answering the following four research questions:

- **RQ1:** How effective is the proposed method in attacking black-box watermarks on different datasets and models?
- **RQ2:** What are the contributions of feature self-distillation and label self-distillation to the overall performance?
- **RQ3:** How does the initial data proportion affect the attack effectiveness and the main-task performance?
- **RQ4:** How does the attention distribution of the model change before and after the watermark attack?

5.1. Experimental Setup

5.1.1. Datasets and Models

We conduct experiments on four datasets: MNIST, CIFAR-10, ImageNet-10, and ImageNet-100. Each dataset is split into 80–20% for training and testing. The specific information of the datasets is shown in Table 1.

Table 1. Basic statistics of the datasets.

Dataset	#Samples	#Classes	#Size
MNIST	70,000	10	28 × 28
CIFAR-10	60,000	10	32 × 32
ImageNet-10	12,000	10	299 × 299
ImageNet-100	120,000	100	299 × 299

In this experiment, four deep learning models are used for experiments and evaluation on different datasets. The MNIST, CIFAR-10, ImageNet-10, and ImageNet-100 datasets adopt VGG16, AlexNet, ResNet50, and DenseNet121, respectively. The specific parameters of these models are shown in Table 2.

Table 2. Basic statistics of the models.

Dataset	Model	Epochs	LR	Accuracy
MNIST	VGG16	100	0.05	93.87%
CIFAR-10	AlexNet	100	0.05	88.92%
ImageNet-10	ResNet50	100	0.02	91.30%
ImageNet-100	DenseNet121	200	0.02	79.92%

5.1.2. Implementation Details

To improve reproducibility, the main implementation settings are specified as follows. The pseudo-label confidence threshold is set to $\tau_{\text{conf}} = 0.95$, and only samples with maximum prediction confidence above this threshold are retained. The feature mask is initialized to 0.5 and optimized using Adam with a learning rate of 0.01 for 500 iterations, with $\lambda_m = 0.01$ and $\rho = 0.5$. The trigger generator is implemented as a lightweight three-layer convolutional network and optimized using Adam with a learning rate of 0.001 for 1000 iterations, subject to the perturbation constraint $|x - G_\theta(x)| \leq \epsilon_{\text{inv}}$ with $\epsilon_{\text{inv}} = 0.05$. For the subsequent removal optimization, the batch size is 64, the contrastive temperature is $\tau_{\text{con}} = 0.07$, and the feature- and label-level self-distillation weights are set to $\lambda_{\text{feat}} = \lambda_{\text{label}} = 1.0$. The trainable target model is optimized using Adam with an initial learning rate of 0.001 for 50 epochs, while the reference model remains frozen throughout training. The learning rate is reduced by a factor of 0.1 after 5 consecutive epochs without improvement, and training is terminated after 10 consecutive epochs without improvement or when the maximum number of epochs is reached. No ground-truth labels are used in any watermark-removal stage; all optimization procedures rely exclusively on the high-confidence pseudo-labeled samples generated from the editable watermarked model.

To reduce the influence of randomness and stochastic optimization on the experimental results, each experiment was independently repeated five times under the same experimental settings. For figures, the reported values were calculated as the average of the results obtained from these five independent runs.

Experimental platform: The experimental environment is based on Python 3.7, with the CPU being I7-7700 and the GPU being RTX 4090, with 16 GB × 4 memory (DDR4). The operating system for development and experiments is Ubuntu 18.04 (OS).

5.1.3. Evaluation Metrics

The evaluation metrics include the watermark verification success rate (VSR) and the main-task accuracy drop (ACC-D). VSR measures the residual watermark behavior and is defined as

$$\mathrm{VSR} = \frac{\mathrm{Ver}_{\mathrm{right}}}{\mathrm{Ver}_{\mathrm{all}}} \times 100\%,$$

where $\mathrm{Ver}_{\mathrm{right}}$ denotes the number of verification samples for which the model produces the designated watermark response, and $\mathrm{Ver}_{\mathrm{all}}$ denotes the total number of verification samples. A lower VSR indicates less residual watermark behavior.

To evaluate the impact of watermark removal on normal task performance, we define

$$\text{ACC-D} = \mathrm{Acc}_{\mathrm{wm}} - \mathrm{Acc}_{\mathrm{rm}},$$

where $\mathrm{Acc}_{\mathrm{wm}}$ and $\mathrm{Acc}_{\mathrm{rm}}$ denote the clean-test classification accuracies, expressed as percentages, of the watermarked model before removal and the corresponding model after removal, respectively. Both accuracies are evaluated on the same clean test set. ACC-D is therefore an absolute accuracy difference expressed in percentage points, rather than a relative percentage decrease.

The pre-removal watermarked model serves as the reference for each experimental setting. Accordingly, its Before Attack ACC-D is 0.00 by definition. A positive ACC-D indicates an accuracy decrease caused by watermark removal, whereas a negative value indicates an accuracy increase. Accuracy changes caused by watermark embedding are outside the scope of this removal-specific metric. When results are aggregated across runs, ACC-D is first calculated within each run using the corresponding pre- and post-removal accuracies and then averaged.

5.1.4. Watermark Embedding Algorithms and Baselines

We adopt four black-box watermarking methods: BadNets [11], WaNet [13], Blend [12], and CTW [14]. For comparison, we evaluate four watermark-removal baselines: model fine-tuning (FT) [20], Fine-Pruning (FP) [4], Neural Cleanse (NC) [21], and Neural Attention Distillation (NAD) [5].

- FT: The editable watermarked model is fine-tuned on the available clean inputs using high-confidence pseudo-labels generated by the pre-removal watermarked model. No ground-truth labels or original watermark samples are supplied.
- FP: Neurons that are inactive on the available clean inputs are pruned according to their activation statistics. The pruned model is subsequently fine-tuned using the same clean inputs and their pseudo-labels, without ground-truth labels or original watermark information.
- NC: The editable model and available clean inputs are used to search for small perturbations associated with candidate target classes. The inferred patterns are then used to construct an input filter for detecting abnormal inputs. The candidate target classes are searched internally; neither the original secret trigger nor the original watermark target labels are provided. This implementation evaluates input filtering rather than direct removal of watermark behavior from the model parameters.
- NAD: A teacher model is obtained by fine-tuning a copy of the watermarked model on the available pseudo-labeled clean inputs. The teacher is then frozen and used to guide the watermarked student model through intermediate-layer attention alignment. Both teacher fine-tuning and student optimization use pseudo-labels rather than ground-truth labels.

Baseline Fairness and Comparison Protocol: All methods are evaluated on the same dataset–model pairs, watermark types, clean-data budgets, and experimental hardware. For each setting, they receive the same initial editable watermarked model, including its architecture and parameters. Intermediate activations and gradients are accessible when

required by the corresponding method. Thus, black-box refers to watermark verification, whereas the removal procedures follow an editable-model threat model.

No method receives ground-truth training labels, original watermark samples, the original secret trigger, or the original watermark target labels. Whenever supervised optimization is required, the method uses the same fixed high-confidence pseudo-labels generated by the pre-removal watermarked model according to Section 5.1.2. Consequently, FT, FP, and NAD are evaluated as unlabeled adaptations of their original formulations. NC uses the available clean inputs and candidate target classes for its internal search.

The original watermark verification set and its associated labels are accessible only to the evaluator and are not used to construct the attacks or the NC filter. Clean-test labels are likewise reserved for evaluation. The optimization procedures use comparable training budgets, while each baseline retains its method-specific objective. For NC, the reported results characterize the model combined with its input filter; they should therefore be distinguished from the parameter-editing results of the other methods.

5.2. Watermark Attack Performance Experiments

In this subsection, the performance of the proposed watermark attack method on different datasets and models is evaluated in detail through comparative experiments to verify its effectiveness and reliability. The experimental results are shown in Figures 2 and 3.

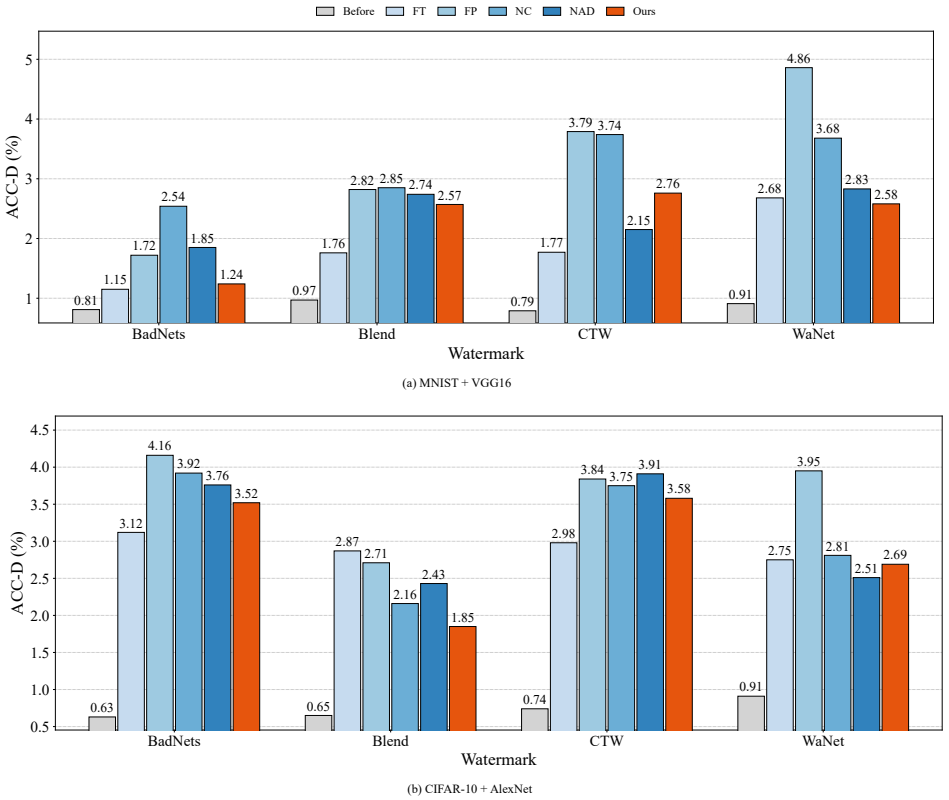


Figure 2. *Cont.*

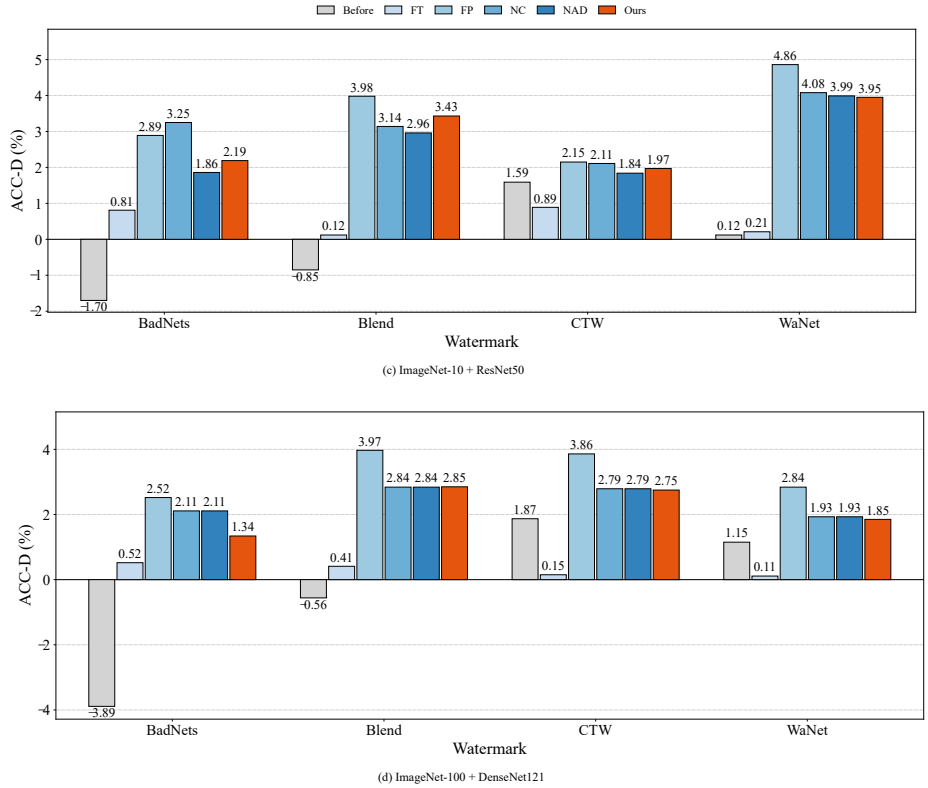


Figure 2. Impact of watermark attack methods on the main-task performance of the model.

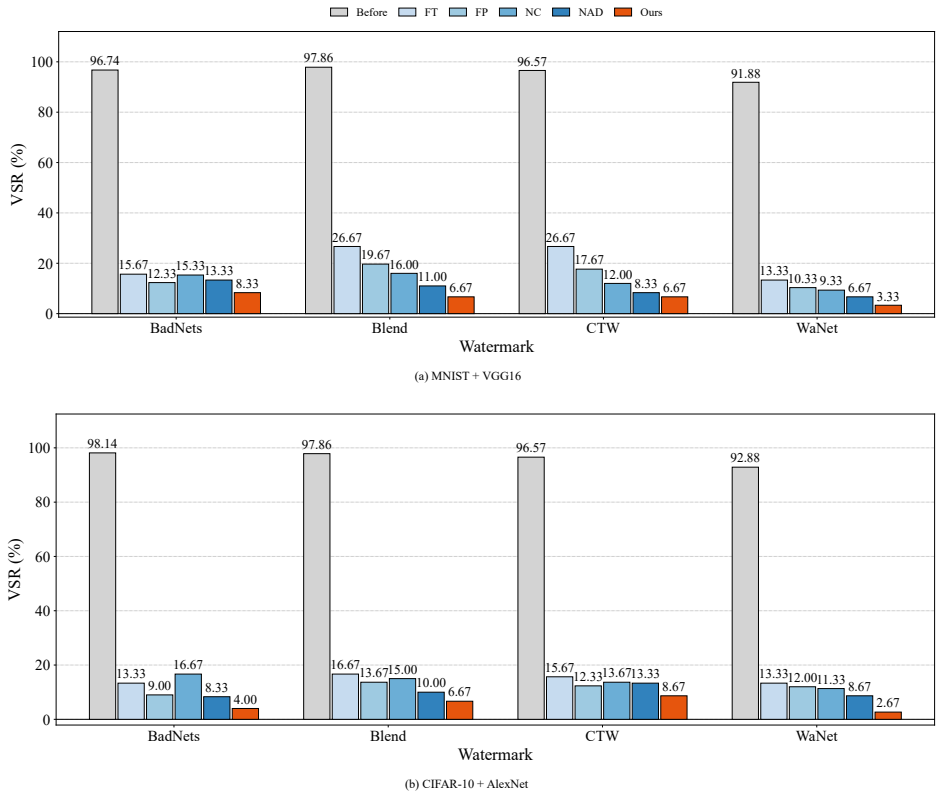


Figure 3. *Cont*.

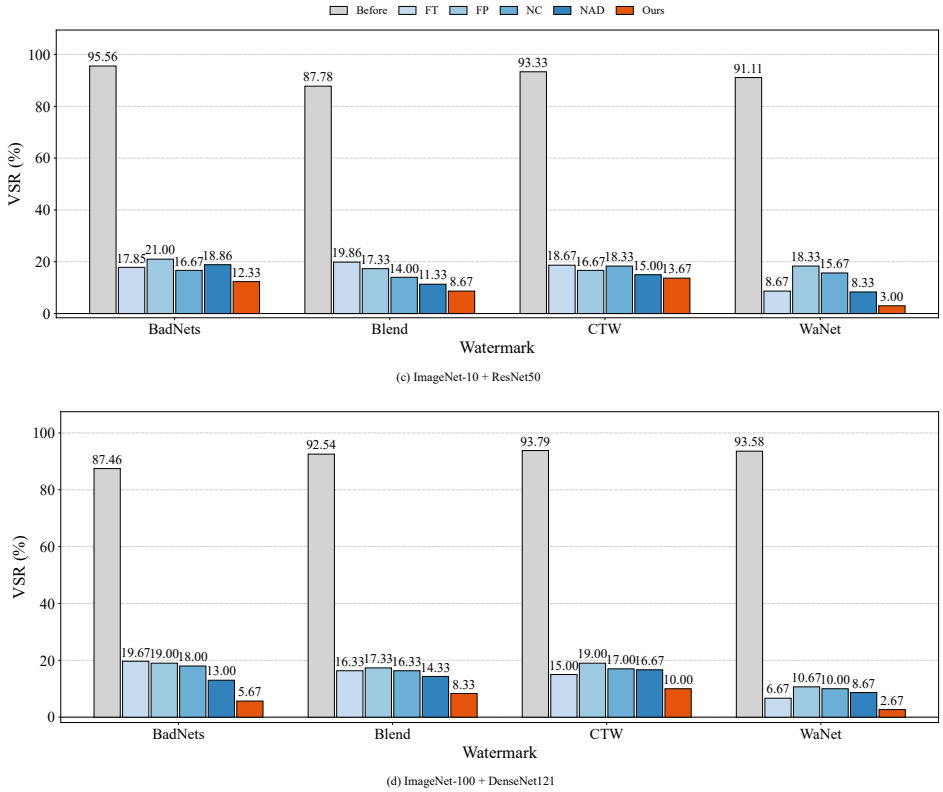


Figure 3. Comparison of the attack performance of different watermark attack methods.

By comparing the performance of different watermark attack methods, including FT, FP, NC, and NAD, it is found that the proposed watermark attack method significantly outperforms the other methods in terms of attack effectiveness. For example, on the MNIST dataset, for the VGG16 model, the VSR of the proposed method under the BadNets watermark is 8.33%, which is lower than those of FT, FP, NC, and NAD by 7.34%, 10%, 20%, and 5%, respectively. This indicates that the proposed method can significantly reduce the watermark verification success rate on different datasets and models, exhibiting a strong watermark attack capability. In addition to the verification success rate, the experiments also focus on the impact of watermark embedding on the main-task performance. The experimental results show that, compared with the other watermark attack methods, the proposed method has the least impact on the main-task performance. This indicates that the proposed method can effectively preserve the original performance of the model while achieving a low watermark verification success rate, minimizing the interference with classification accuracy.

In summary, the proposed method, a self-distillation watermark attack technique based on contrastive learning, introduces a contrastive learning mechanism and a self-distillation strategy to remove the watermark while preserving the original classification capability of the model to the greatest extent. The core idea of this method is to use contrastive learning to enhance the ability of the model to extract target features while weakening its dependence on watermark features, thereby effectively separating the watermark information from the target features. On this basis, through the self-distillation process, the model gradually realizes an all-round watermark attack from shallow to deep layers and from the feature level to the label level, ensuring that the watermark information is thoroughly removed while the ability of the model to recognize target features is fully preserved. This not only improves the accuracy of the watermark attack, but also significantly enhances the robustness of the model, enabling it to maintain excellent performance in complex tasks.

5.3. Ablation Experiments

In this experiment, the proposed watermark attack method is analyzed in greater detail to evaluate the necessity and impact of each key component. In addition to feature- and label-level self-distillation, we further examine the contributions of confidence-filtered pseudo-label generation, feature decoupling, trigger inversion, and contrastive learning. By removing individual components while keeping the remaining components unchanged, we investigate their respective roles in watermark removal.

Figure 4 presents the component-wise ablation results on CIFAR-10 with AlexNet under four representative watermark schemes. The complete method consistently achieves the lowest residual VSR and a relatively low ACC-D. Removing any individual component leads to a degradation in watermark-removal effectiveness and/or main-task preservation. In particular, removing contrastive learning causes a clear increase in residual VSR, demonstrating its important role in driving the model away from watermark-related representations. Removing feature decoupling and trigger inversion also noticeably weakens the attack, indicating that identifying and exposing watermark-sensitive features is important for the subsequent removal optimization. The degradation caused by removing pseudo-label generation further verifies the importance of reliable semantic supervision in the unlabeled setting.

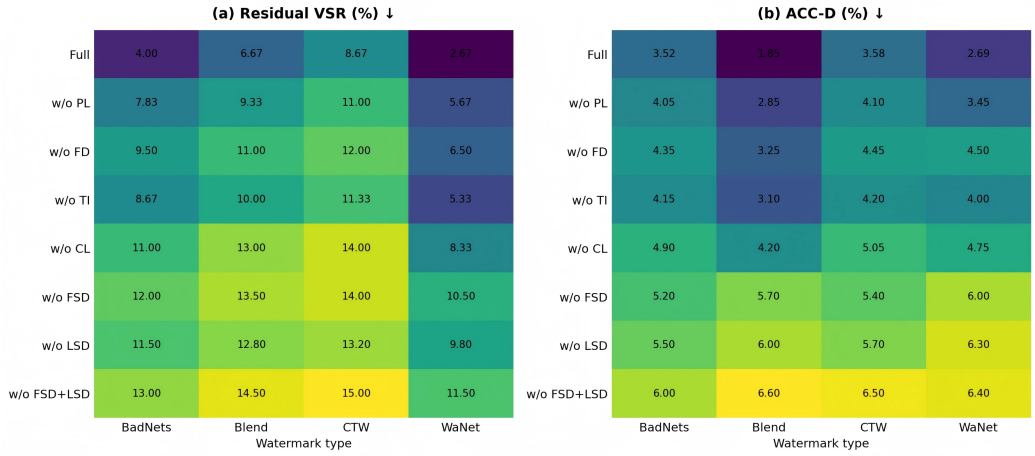


Figure 4. Component-wise ablation results on CIFAR-10 with AlexNet. Lower VSR and ACC-D indicate better performance. PL, FD, TI, CL, FSD, and LSD denote pseudo-label generation, feature decoupling, trigger inversion, contrastive learning, feature self-distillation, and label self-distillation, respectively.

The above observations are complementary to the self-distillation ablation results in Table 3. When both feature self-distillation and label self-distillation are missing, the watermark attack performance drops significantly, while the impact on the main-task performance becomes the largest. For example, on the MNIST dataset, the VSR of the VGG16 model under the WaNet watermark increases from 3.33% with the full method to 12.33%, while ACC-D increases from 2.58% to 6.50%. Similar trends are observed across the other datasets and watermark schemes, indicating that the two self-distillation mechanisms jointly contribute to effective watermark removal and task preservation.

When only feature self-distillation or label self-distillation is removed, the degradation is smaller than that caused by removing both mechanisms. For example, on CIFAR-10 with AlexNet under BadNets, the VSR increases from 4.00% with the full method to 12.00% and 11.50% when feature self-distillation and label self-distillation are removed, respectively, and further increases to 13.00% when both are removed. These results demonstrate that feature self-distillation and label self-distillation provide complementary constraints at the feature and label levels.

Table 3. Ablation study of the self-distillation components of the proposed watermark attack. Values are reported as VSR (%)/ACC-D (%). F = Feature, L = Label, SD = Self-distillation.

Dataset	Model	Watermark	VSR (%)/ACC-D (%)				
			Before Attack	w/o F&L SD	w/o F SD	w/o L SD	Ours
MNIST	VGG16	BadNets	96.74/0.00	15.33/5.50	14.00/4.80	13.50/5.20	8.33/1.24
		Blend	97.86/0.00	14.67/6.20	13.50/5.60	12.80/5.90	6.67/2.57
		CTW	96.57/0.00	14.00/5.80	13.20/5.10	12.50/5.40	6.67/2.76
		WaNet	99.18/0.00	12.33/6.50	11.50/5.90	10.80/6.20	3.33/2.58
CIFAR-10	AlexNet	BadNets	98.14/0.00	13.00/6.00	12.00/5.20	11.50/5.50	4.00/3.52
		Blend	97.86/0.00	14.50/6.50	13.50/5.70	12.80/6.00	6.67/1.85
		CTW	96.57/0.00	15.00/6.20	14.00/5.40	13.20/5.70	8.67/3.58
		WaNet	98.76/0.00	11.50/6.80	10.50/6.00	9.80/6.30	2.67/2.69
ImageNet-10	ResNet50	BadNets	95.56/0.00	18.33/7.50	17.00/6.80	16.50/7.10	12.33/2.19
		Blend	87.78/0.00	16.50/7.20	15.50/6.50	14.80/6.80	8.67/3.43
		CTW	93.33/0.00	19.00/7.80	18.00/7.10	17.20/7.40	13.67/1.97
		WaNet	97.78/0.00	12.00/8.00	11.00/7.30	10.50/7.60	3.00/3.95
ImageNet-100	DenseNet121	BadNets	87.46/0.00	14.00/8.50	13.00/7.80	12.50/8.10	5.67/1.34
		Blend	92.54/0.00	16.00/8.20	15.00/7.50	14.50/7.80	8.33/2.85
		CTW	93.79/0.00	17.00/8.80	16.00/8.10	15.50/8.40	10.00/2.75
		WaNet	93.58/0.00	11.50/8.30	10.50/7.60	9.80/7.90	2.67/1.85

Overall, the component-wise ablation confirms that the proposed modules play complementary roles in the watermark-removal process. Pseudo-label generation provides reliable supervision, feature decoupling identifies watermark-related representations, trigger inversion constructs watermark-sensitive perturbations, contrastive learning drives the representation away from watermark-related features, and self-distillation preserves task-relevant information at both feature and label levels. Their combination enables the proposed method to achieve effective watermark removal while maintaining the main-task performance under limited-data and unlabeled conditions.

5.4. Parameter Sensitivity Analysis

We investigate the influence of the initial data proportion on watermark-removal effectiveness and main-task accuracy using ImageNet-10 with ResNet50. Table 4 reports the results for initial data proportions of 5%, 10%, 20%, and 30%. ACC-D is measured relative to the corresponding pre-removal watermarked model, whose Before Attack ACC-D is 0.00.

Table 4. Sensitivity analysis of the initial data proportion on ImageNet-10(ResNet50).

Watermark	Metric	Before	5%	10%	20%	30%
BadNets	ACC-D	0.00%	2.74%	2.19%	2.19%	1.12%
	VSR	95.56%	18.00%	12.33%	8.33%	2.00%
Blend	ACC-D	0.00%	3.99%	3.43%	2.82%	1.58%
	VSR	87.78%	21.33%	8.67%	6.67%	3.33%
CTW	ACC-D	0.00%	2.84%	1.97%	1.29%	0.86%
	VSR	93.33%	16.00%	13.67%	8.67%	4.67%
WaNet	ACC-D	0.00%	4.98%	3.95%	3.51%	2.13%
	VSR	97.78%	10.33%	3.00%	3.00%	0.00%

Increasing the initial data proportion generally reduces both residual VSR and ACC-D. For BadNets, VSR decreases from 18.00% at a 5% data proportion to 2.00% at 30%, while ACC-D decreases from 2.74 to 1.12 percentage points. Similar trends are observed for Blend

and CTW. For WaNet, VSR decreases from 10.33% to 0.00%, while ACC-D decreases from 4.98 to 2.13 percentage points.

The largest reported mean ACC-D in this sensitivity analysis is 4.98 percentage points, observed for WaNet at the 5% data proportion. This exceeds the maximum of 3.95 percentage points for the complete method in the main comparison. Accordingly, 4.98 percentage points is the largest reported mean accuracy drop when both the main comparison and the data-proportion sensitivity experiments are considered. This statement concerns the complete method and does not include variants with components removed in the ablation study.

The results also show that additional data can improve watermark removal without increasing accuracy degradation. For example, increasing the data proportion from 10% to 20% reduces the BadNets VSR from 12.33% to 8.33%, while ACC-D remains at 2.19 percentage points. However, a 20% data proportion does not yield the lowest VSR or ACC-D among all tested settings. The choice of data proportion should therefore reflect the available clean-data budget and the desired removal effectiveness.

5.5. Computational Cost Analysis

We further evaluate the computational cost of the proposed attack on MNIST with VGG16. Table 5 reports the attack time, model queries, and peak GPU memory. As shown in Table 5, our method requires slightly more time and memory than FT, FP, and NAD, mainly due to the feature-mask optimization, trigger inversion, and self-distillation. However, its computational cost remains comparable to the baselines, while requiring only one model query per image for pseudo-label generation. The peak memory usage of 3.2 GB is also well within the capacity of the GPU used in our experiments. These results indicate that the additional computational cost is moderate and acceptable for achieving stronger watermark removal.

Table 5. Computational cost comparison on MNIST + VGG16.

Method	Time (s/Image)	Queries/Image	Memory (GB)
FT	0.22	20	1.8
FP	0.24	20	1.7
NC	0.47	50	2.4
NAD	0.29	30	2.8
Ours	0.36	1	3.2

5.6. Visualization of Feature Decoupling and Trigger Inversion

Figure 5 illustrates feature decoupling and trigger inference through the learned feature mask, the inferred watermark-related perturbation, and the corresponding clean and perturbed inputs. The mask separates retained and complementary features, while the inferred perturbation guides watermark removal without requiring exact recovery of the secret trigger.

To assess the contribution of perturbation inference, we compare inferred and random perturbations on CIFAR-10 with AlexNet under BadNets, Blend, CTW, and WaNet. Only the perturbation source varies; all other removal settings are held fixed, with perturbation norms after input clipping. The original watermark verification set is used matched exclusively for evaluation. Residual VSR and ACC-D should be reported as the mean and sample standard deviation over five paired runs.

The illustrative results in Table 6 show lower residual VSR with inferred perturbations across all four watermark types. For BadNets, residual VSR decreases from 14.00% to 4.00%, with comparable ACC-D values of 3.60 and 3.52 percentage points, respectively. The corresponding VSR reductions for Blend, CTW, and WaNet are 9.33, 10.33, and 7.33 percentage points. These results illustrate the potential benefit of model-derived perturbations for watermark removal; empirical validation is needed to establish their effectiveness.

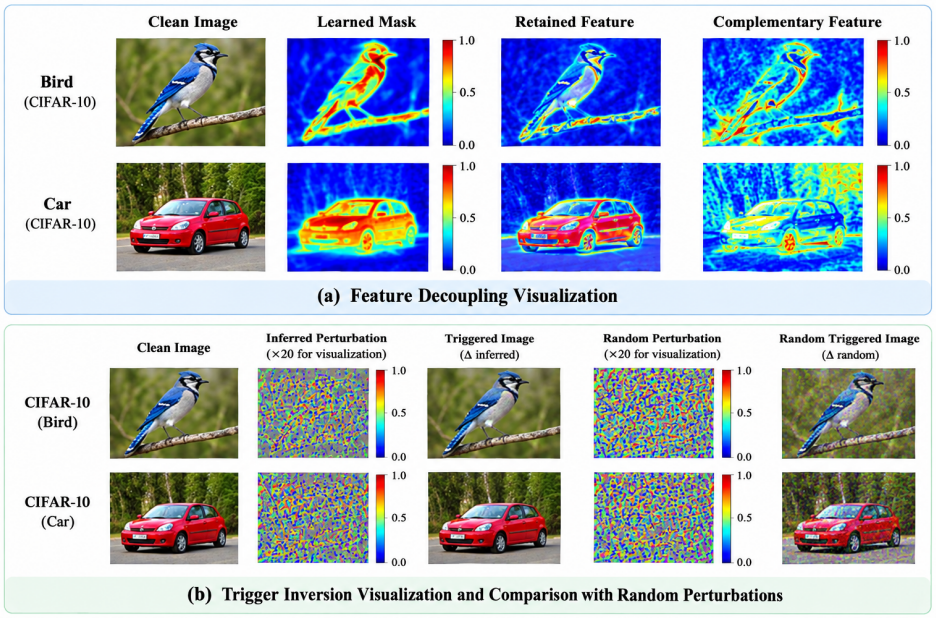


Figure 5. Visualization of feature decoupling and trigger inversion. The learned mask separates retained and complementary features, while the inferred perturbation is compared with a random perturbation of comparable magnitude.

Table 6. Comparison of inferred and random perturbations on CIFAR-10 with AlexNet.

Watermark	Perturbation	VSR ↓	ACC-D ↓
BadNets	Random	14.00 ± 0.95	3.60 ± 0.16
	Inferred	4.00 ± 0.47	3.52 ± 0.13
Blend	Random	16.00 ± 0.95	2.10 ± 0.16
	Inferred	6.67 ± 0.47	1.85 ± 0.13
CTW	Random	19.00 ± 0.95	3.80 ± 0.16
	Inferred	8.67 ± 0.47	3.58 ± 0.13
WaNet	Random	10.00 ± 0.95	2.90 ± 0.16
	Inferred	2.67 ± 0.47	2.69 ± 0.13

5.7. Interpretability Experiments

In deep learning research, interpretability experiments are of important theoretical significance and practical value. First, through visualization techniques, feature importance analysis, and attention mechanisms, interpretability experiments reveal the internal decision logic of the model, thereby alleviating the "black box" problem caused by the highly nonlinear structure of deep learning models. In this subsection, the attention heatmaps of the watermarked model before and after watermark removal are visualized to analyze the watermark attack effect.

As shown in Figure 6, the watermarked model exhibits a significant feature shift in the distribution of the attention heatmaps. Specifically, for watermark labels embedded in the form of a patch, the attention of the model is highly concentrated on the local region where the watermark is located, leading to insufficient attention to the main features of the target object in the image. This attention concentration indicates that the watermark label is treated by the model as an important discriminative feature during training, which interferes with the normal learning of the target object. For watermarks in the form of global perturbation, the attention distribution of the model shows an obvious divergent trend: the attention shifts from the target object to the background region, or the attention

intensity on the key features of the target is significantly weakened. Such dispersion or weakening of attention directly causes deviations in the decision-making process of the model, making it difficult to accurately capture the essential features of the target object, thereby leading to classification errors or degraded recognition performance. However, after the watermark attack, the attention distribution of the model is significantly improved: the attention refocuses on the key feature regions of the target object, such as its contour, texture, or salient local features, thereby restoring the ability of the model to correctly recognize the target object and achieving accurate classification decisions.

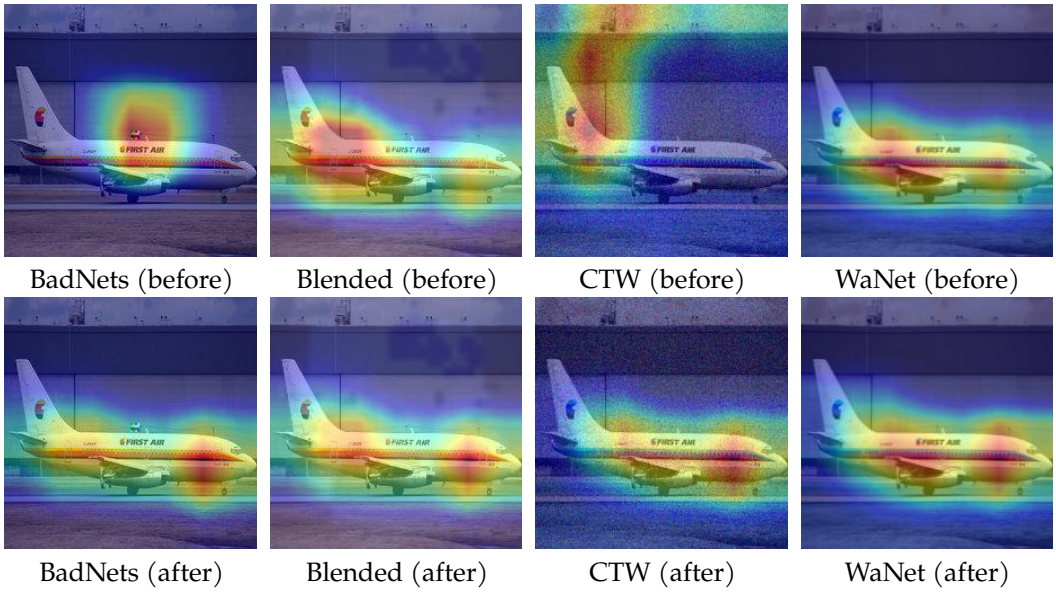


Figure 6. Changes in the model attention heatmaps before and after the watermark attack (from **left** to **right**, the watermark embedding methods are BadNets, Blended, CTW, and WaNet).

6. Conclusions

This paper proposed a self-distillation watermark attack method based on contrastive learning. High-reliability pseudo-labels are generated through data augmentation and confidence-based filtering. Combined with feature decoupling, an approximate watermark-related perturbation is inferred without requiring exact recovery of the original secret trigger. By further integrating contrastive learning with feature- and label-level self-distillation, the proposed method effectively suppresses watermark-related representations while preserving normal task-relevant features, enabling effective watermark removal in the limited-data, unlabeled setting.

We conducted a comprehensive experimental evaluation covering watermark-removal effectiveness, component-wise ablation, parameter sensitivity, and interpretability. The results show that the proposed method reduces residual watermark behavior while largely maintaining main-task performance. On CIFAR-10 and ImageNet-100, the method reduces the verification success rate of the BadNets watermark to 4.0% and 5.67%, respectively. For the complete method, the largest reported mean accuracy drop relative to the pre-removal watermarked model is 3.95 percentage points in the main comparison. When the data-proportion sensitivity experiments are included, the largest reported mean drop is 4.98 percentage points, observed for WaNet on ImageNet-10 with ResNet50 at a 5% initial data proportion. The ablation experiments support the complementary contributions of the proposed components, while the sensitivity analysis shows that increasing the initial data proportion generally improves watermark-removal effectiveness and reduces accuracy degradation. Finally, the visualization results show that, after watermark removal, the

model's attention tends to refocus on task-relevant regions, providing qualitative support for the proposed mechanism.

Nevertheless, this work still has some limitations. The proposed method is mainly evaluated on image classification tasks with four representative black-box watermarking schemes, and its applicability to other tasks and more recently proposed watermarking schemes remains to be investigated. In future work, we plan to study adaptive watermark attacks against more robust watermarking schemes and further reduce the dependence on the amount of available clean data.

Author Contributions: Conceptualization, W.W. and S.Y.; methodology, W.W., S.Y. and J.Z.; software, W.W. and J.Z.; validation, W.W., J.Z. and S.Z.; formal analysis, W.W., S.Y. and J.C.; investigation, W.W., J.Z., S.Z. and Y.B.; resources, S.Y. and J.C.; data curation, W.W. and J.Z.; writing—original draft preparation, W.W.; writing—review and editing, S.Y., J.Z., S.Z., Y.B. and J.C.; visualization, W.W. and S.Z.; supervision, S.Y. and J.C.; project administration, S.Y. and J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Zhejiang Provincial Natural Science Foundation (LDQ23F020001), National Natural Science Foundation of China (62072406), and Key Laboratory of the Fifth Research Institute of Electronics, Ministry of Industry and Information Technology (No. HK00202503455).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M.K.; Ristenpart, T. Stealing machine learning models via prediction {APIs}. In Proceedings of the 25th USENIX security symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 601–618.
2. Zhang, C.; Luo, S.; Pan, L.; Gu, D.; Yuan, J. A new model stealing defense based on DNN retraining for decision boundary protection. *Neurocomputing* **2026**, *693*, 133816. [CrossRef]
3. Li, Y.; Wang, H.; Barni, M. A survey of Deep Neural Network watermarking techniques. *Neurocomputing* **2021**, *461*, 171–193. [CrossRef]
4. Liu, K.; Dolan-Gavitt, B.; Garg, S. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In *Research in Attacks, Intrusions, and Defenses*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; pp. 273–294. [CrossRef]
5. Li, Y.; Lyu, X.; Koren, N.; Lyu, L.; Li, B.; Ma, X. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. In Proceedings of the International Conference on Learning Representations, Online, 3–7 May 2021.
6. Uchida, Y.; Nagai, Y.; Sakazawa, S.; Satoh, S. Embedding Watermarks into Deep Neural Networks. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, Bucharest, Romania, 6–9 June 2017; pp. 269–277. [CrossRef]
7. Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M.P.; Huang, H.; Molloy, I. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Republic of Korea, 4–8 June 2018; pp. 159–172. [CrossRef]
8. Darvish Rouhani, B.; Chen, H.; Koushanfar, F. DeepSigns. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, Providence, RI, USA, 13–17 April 2019; pp. 485–497. [CrossRef]
9. Li, Q.; Xu, G. Deep neural network fingerprinting by general examples. *Neurocomputing* **2026**, *665*, 132184. [CrossRef]
10. Wei, J.; Zhou, J.; Men, Y.; Luo, S.; Pan, L. High-fidelity backdoor watermark embedding framework for classification models in heterogeneous tabular data. *Neurocomputing* **2026**, *687*, 133646. [CrossRef]
11. Gu, T.; Dolan-Gavitt, B.; Garg, S. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *arXiv* **2017**, arXiv:1708.06733. [CrossRef]
12. 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. [CrossRef]
13. Nguyen, A.; Tran, A. WaNet—Imperceptible Warping-based Backdoor Attack. *arXiv* **2021**, arXiv:2102.10369. [CrossRef]
14. Bansal, A.; Chiang, P.y.; Curry, M.J.; Jain, R.; Wigington, C.; Manjunatha, V.; Dickerson, J.P.; Goldstein, T. Certified neural network watermarks with randomized smoothing. *Proc. Int. Conf. Mach. Learn.* **2022**, *162*, 1450–1465.

15. 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), Baltimore, MD, USA, 15–17 August 2018; pp. 1615–1631.
16. Li, M.; Wu, H.; Zhang, X. A novel watermarking framework for intellectual property protection of NLG APIs. *Neurocomputing* **2023**, *558*, 126700. [CrossRef]
17. Hu, Z.; Zhu, J.; Zhang, B.; Hu, X. Amplification trojan network: Attack deep neural networks by amplifying their inherent weakness. *Neurocomputing* **2022**, *505*, 142–153. [CrossRef]
18. Chen, X.; Li, M.; Sun, Y.; Tian, Z. Invisible trigger image: A dynamic neural backdoor attack based on hidden feature. *Neurocomputing* **2025**, *639*, 130296. [CrossRef]
19. Dai, J.; Xiong, Z.; Cao, C. A semantic backdoor attack against graph convolutional networks. *Neurocomputing* **2024**, *600*, 128133. [CrossRef]
20. Aiken, W.; Kim, H.; Woo, S.; Ryoo, J. Neural network laundering: Removing black-box backdoor watermarks from deep neural networks. *Comput. Secur.* **2021**, *106*, 102277. [CrossRef]
21. Wang, B.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; Zhao, B.Y. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 707–723. [CrossRef]
22. Xiang, Z.; Miller, D.J.; Kesidis, G. Reverse engineering imperceptible backdoor attacks on deep neural networks for detection and training set cleansing. *Comput. Secur.* **2021**, *106*, 102280. [CrossRef]
23. Li, P.; Huang, J.; Wu, H.; Zhang, Z.; Qi, C. SecureNet: Proactive intellectual property protection and model security defense for DNNs based on backdoor learning. *Neural Netw.* **2024**, *174*, 106199. [CrossRef]
24. Zhang, F.; Xu, Y.; Lv, J.; Yu, Z.; Zhang, L.; Feng, J.; Liu, X. Backdoor-vaccine: An active backdoor defense framework via competitive backdoor injection. *Neurocomputing* **2026**, *664*, 132133. [CrossRef]
25. Miah, A.A.; Bi, Y. DeepCurer: Pruning-based backdoor mitigation via progressive neuron ranking using adversarial proxies. *Neurocomputing* **2026**, *699*, 134409. [CrossRef]
26. Wu, C.; Ren, X.; Liu, C.; Xue, Y. NeuroPatch: Lightweight diffusion model repair for backdoor attack mitigation based on neuron-level patching. *Neurocomputing* **2026**, *680*, 133283. [CrossRef]
27. Wei, Y.; Gao, H.; Wang, Y.; Gao, Y.; Liu, H. A lightweight backdoor defense framework based on image inpainting. *Neurocomputing* **2023**, *537*, 22–36. [CrossRef]
28. Chen, C.; Dai, J. Mitigating backdoor attacks in LSTM-based text classification systems by Backdoor Keyword Identification. *Neurocomputing* **2021**, *452*, 253–262. [CrossRef]
29. Li, Y.; Wang, Y.; Yuan, H.; Zhang, P.; Huang, W.; Liu, Z. Backdoor defense framework with sparse training and detection in federated learning. *Neurocomputing* **2026**, *661*, 131831. [CrossRef]
30. Zhang, A.; Chen, H.; Wang, X.; Li, J.; Gao, Y.; Wang, X. Defending against backdoor attack on deep neural networks based on multi-scale inactivation. *Inf. Sci.* **2025**, *690*, 121562. [CrossRef]
31. Huang, L.; Zhang, L.Y.; Chang, C.C.; Wang, W.; Qin, C. Backdoor defense based on adversarial prediction proximity and contrastive knowledge distillation. *Pattern Recognit.* **2026**, *172*, 112336. [CrossRef]
32. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. *Proc. Int. Conf. Mach. Learn.* **2020**, *119*, 1597–1607.
33. AlSuwat, M.; Al-Shareef, S.; AlGhamdi, M. Audio–visual self-supervised representation learning: A survey. *Neurocomputing* **2025**, *634*, 129750. [CrossRef]
34. Beyer, L.; Zhai, X.; Royer, A.; Markeeva, L.; Anil, R.; Kolesnikov, A. Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: Piscataway, NJ, USA, 2022; pp. 10915–10924.
35. Tzelepi, M.; Passalis, N.; Tefas, A. Probabilistic online self-distillation. *Neurocomputing* **2022**, *493*, 592–604. [CrossRef]
36. Lu, S.; Zeng, W.; Li, X.; Ou, J. Adaptive lightweight network construction method for Self-Knowledge Distillation. *Neurocomputing* **2025**, *624*, 129477. [CrossRef]
37. Yang, C.; Lu, Y.; Qiu, Y.; Wang, Y. IKFD-Depth: Enhancing self-supervised monocular depth estimation with incremental knowledge distillation and feature decoupling self-distillation. *Neurocomputing* **2025**, *640*, 130314. [CrossRef]
38. Wang, Q.; Zhou, J. Top-KD: Top-scaled contrastive knowledge distillation. *Neurocomputing* **2026**, *685*, 133636. [CrossRef]

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