NEURAL DEHYDRATION: Effective Erasure of Black-box Watermarks from DNNs with Limited Data

Yifan Lu
Fudan University
Shanghai, China
luyf23@m.fudan.edu.cn

Wenxuan Li
Fudan University
Shanghai, China
liwx22@m.fudan.edu.cn

Mi Zhang*
Fudan University
Shanghai, China
mi_zhang@fudan.edu.cn

Xudong Pan
Fudan University
Shanghai, China
xdpan@fudan.edu.cn

Min Yang*
Fudan University
Shanghai, China
m_yang@fudan.edu.cn

ABSTRACT

To protect the intellectual property of well-trained deep neural networks (DNNs), black-box watermarks, which are embedded into the prediction behavior of DNN models on a set of specially-crafted samples and extracted from suspect models using only API access, have gained increasing popularity in both academy and industry. Watermark robustness is usually implemented against attackers who steal the protected model and obfuscate its parameters for watermark removal. However, current robustness evaluations are primarily performed under moderate attacks or unrealistic settings. Existing removal attacks could only crack a small subset of the mainstream black-box watermarks, and fall short in four key aspects: incomplete removal, reliance on prior knowledge of the watermark, performance degradation, and high dependency on data.

In this paper, we propose a watermark-agnostic removal attack called NEURAL DEHYDRATION (*abbrev.* DEHYDRA), which effectively erases all ten mainstream black-box watermarks from DNNs, with only limited or even no data dependence. In general, our attack pipeline exploits the internals of the protected model to recover and unlearn the watermark message. We further design target class detection and recovered sample splitting algorithms to reduce the utility loss and achieve data-free watermark removal on five of the watermarking schemes. We conduct comprehensive evaluation of DEHYDRA against ten mainstream black-box watermarks on three benchmark datasets and DNN architectures. Compared with existing removal attacks, DEHYDRA achieves strong removal effectiveness across all the covered watermarks, preserving at least 90% of the stolen model utility, under the data-limited settings, i.e., less than 2% of the training data or even data-free. Our work reveals the vulnerabilities of existing black-box DNN watermarks in realistic settings, highlighting the urgent need for more robust watermarking techniques. To facilitate future studies, we open-source our code in the following repository: https://github.com/LouisVann/Dehydra.

*Corresponding authors: Mi Zhang and Min Yang.

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690334

CCS CONCEPTS

• **Security and privacy → Security services**; • **Computing methodologies → Artificial intelligence**.

KEYWORDS

Model Watermarking; Robustness; Removal Attack

ACM Reference Format:
Yifan Lu, Wenxuan Li, Mi Zhang, Xudong Pan, and Min Yang. 2024. NEURAL DEHYDRATION: Effective Erasure of Black-box Watermarks from DNNs with Limited Data. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3658644.3690334

1 INTRODUCTION

In recent years, deep neural networks (DNNs) are empowering real-world applications in computer vision [46, 64], natural language processing [16, 22], and autonomous driving [4, 5]. However, training a modern DNN from scratch requires time-consuming data collection and high-end computing resources. To protect the intellectual property of well-trained DNNs, *model watermarking* is an emerging tool for verifying the ownership of DNNs in case of model stealing or abusing.

Generally, in a DNN watermarking scheme, the secret watermark information is embedded into the *target model* at the watermark embedding stage during training. At the subsequent verification stage, the watermark can be extracted from the *suspect model* to determine its ownership. Depending on how the suspect model is accessed during verification, current watermark algorithms can be categorized into *white-box* and *black-box* watermarks. A white-box watermark is usually embedded into the suspect model's internal parameters [9, 54, 57] or neuron activation [15]. In comparison, a black-box watermark is embedded into a model's prediction behaviour on a set of specially-crafted samples (i.e., *watermark data*) by specifying their expected classification results (i.e., *target classes*) [2, 27, 34, 63]. Owing to the weaker access requirement for verification, black-box watermarks have gained increasing popularity in both academia and industry (e.g., IBM [19]), with various designs in watermark data and target class settings spurred.

The robustness of black-box DNN watermarking schemes against *removal attacks*, which modifies the model parameters to cause verification failure, is crucial for their real-world trustworthiness. Unfortunately, current robustness evaluations are mostly performed under moderate attacks or unrealistic settings. Specifically, existing attacks are mainly categorized into three types: pruning-based [38, 54], finetuning-based [10, 47, 65] and unlearning-based [3]. However, these approaches could only crack a small subset of existing black-box watermarks (i.e., reducing the watermark verification success rate to almost random, with most of the utility preserved), which are unable to pose practical threats when the attacker is watermark-agnostic. To be more specific, these attacks may to varied degrees remove the watermark incompletely, require prior knowledge of the target watermark, hamper the model's utility or have a strong dependence on the clean dataset. Recently, another type of model extraction-based attack has been demonstrated to effectively remove the watermarks when over 30% of the training data is available [42, 53]. However, as commercial DNNs are usually trained on private data, it is usually impractical to access such an amount of training data. A general comparison of their attack budgets is summarized in Table 1 and a more detailed analysis is presented in Section 2.2.

Our Work. In this paper, we propose a watermark-agnostic removal attack called NEURAL DEHYDRATION (*abbrev.* DEHYDRA), which effectively erases all ten mainstream black-box watermarks from DNNs, with only limited or even no data dependence. To achieve these objectives, we mainly address the following two challenges:

C1: How to design a watermark-agnostic effective removal attack?

Despite the complex designs of existing black-box watermarks in watermark data and target class settings, we identify a shared trait: they exploit the over-parameterization of DNNs to especially memorize the watermark data correlated to the target classes. Therefore, the model internals should be a sufficient source to reconstruct the watermark information. Inspired by this, we design a general-purpose attack framework to **recover and unlearn** the watermark data from the protected model, similar to the dehydration reaction in chemistry. Specifically, at the first stage, we use an aggressive model inversion technique to reverse-engineer class-wise samples close to the real watermark data from a target watermarked model (§4.1). At the second stage, we intentionally unlearn these samples during the finetuning process (§4.2). This basic DEHYDRA framework is effective against all the ten investigated black-box watermarks.

C2: How to preserve the model utility and reduce the data dependence during attack?

Despite effective, the basic attack might compromise the model's utility, because the recovered samples could also contain information critical to the main task. To address this issue, we further enhance DEHYDRA in the following directions. ❶ We improve the recovering stage of DEHYDRA via incorporating **target class detection**. For watermarks with fixed target classes (such as [63] and [27], where all the watermark data are paired with the identical target class), we derive an observation called *watermark smoothness discrepancy* that, *the loss landscape of model is smoother on the target class than other classes.* We provide both theoretical and empirical analysis on the class-wise smoothness properties, and leverage the discrepancy to distinguish whether the target watermark has a fixed target class, and, if so, detect its target class. This allows us to specially unlearn the recovered samples belonging to the target class when cracking a fixed-class watermark (§5.1). ❷ We improve the unlearning stage of DEHYDRA via **splitting the recovered samples**. Our design are based on the observed *normal data dominance* phenomenon that, *the class-wise recovered samples should be closer to normal samples compared with the watermark samples in that class.* Based on this, we develop a neuron-level criterion to carefully split the recovered samples into proxy watermark data and proxy normal data, with the proxy watermark data to ensure the removal effectiveness and the proxy normal data to preserve the original utility (§5.2). With the above improvements, DEHYDRA minimally impacts the model's utility while ensuring its watermark-agnostic removal effectiveness. Moreover, the split proxy normal data can even allow for a data-free removal attack for watermarks identified with fixed target classes. Table 1 summarizes the advantages of our attack compared with existing removal approaches.

Table 1: Comparison of attack budgets between our attack and existing watermark removal attacks, where ●/◐/○ represent high, medium and low (or no) attack budget.

Attack Type	Removal Ineffectiveness	Watermark Knowledge	Utility Loss	Dataset Access
Prune	◐	○	◐	○
Finetune	◐	○	◐	●
Unlearn	○	●	○	◐
DEHYDRA	○	○	○	○

We conduct comprehensive evaluation of DEHYDRA against ten mainstream black-box watermarks on three benchmark datasets and DNN architectures, under three different data settings. Compared with six baseline removal attacks, our proposed DEHYDRA achieves strong removal effectiveness across all the covered watermarks, preserving at least 90% model utility, under the data-limited settings, i.e., less than 2% of the training data or even data-free.

To facilitate future studies, we open-source our code in https://github.com/LouisVann/Dehydra. Also, an extended version of this paper, with appendix included, is available in [40].

Our Contribution. We summarize the key contributions of this work as follows:

- We propose a watermark-agnostic removal framework, namely DEHYDRA, which exploits the model internals to recover and unlearn the underlying watermarks.
- Based on in-depth analysis, theoretical justifications and pilot studies, we further improve DEHYDRA with target class detection and recovered sample splitting algorithms, which help preserve the model utility and significantly reduce the dataset dependence.
- Extensive evaluations on various settings against ten mainstream black-box watermarks show that DEHYDRA is generally effective and has minimal utility impact, with much more relaxed or even no assumptions on the dataset availability.

2 BACKGROUND

2.1 DNN Watermarks

Traditional model protection techniques employ proactive strategies to prevent unauthorized access, such as encrypting model parameters or restricting access to APIs. However, these methods have been proven by subsequent research to face leakage risks [50, 53]. To further claim and trace the ownership of DNN models, model watermarking technology has emerged.

A white-box watermarking scheme typically embeds a sequence of secret messages into the parameters or neural activations of the target model, and thus requires white-box access of the suspect model to extract the watermark [9, 15, 54, 57]. On the contrary, a black-box watermark (sometimes also called backdoor-based [39, 47], poisoning-based [21] or trigger set-based [34] watermarks) is embedded in the model's prediction behavior on a set of specially-crafted secret samples (i.e., *watermark data*) by specifying their expected prediction results (i.e., *target classes*) [2, 27, 30, 63]. Therefore, the verification process of black-box watermarking only requires the access to the prediction API.

To build a trustworthy ownership verification process, an ideal watermarking scheme should satisfy at least the following three key requirements [42, 59]. ❶ **Fidelity:** The performance degradation on the target model should be as low as possible after the watermark is embedded. ❷ **Integrity:** Models which are trained independently without access to the target model should not be verified as containing the watermark. ❸ **Robustness:** The embedded watermark in the target model should be resistant to potential removal attacks. In the worst cases when the watermark is removed, the utility of the model should decrease dramatically.

We mainly study on the black-box watermarks due to their more realistic settings. Existing black-box watermark schemes have diverse designs in the choices of the *watermark data* and the *target label*. In terms of watermark data, some work leverages specially-crafted patterns [27, 63] or random noise [63] as watermark patterns. Other black-box watermarks may use out-of-distribution samples as watermark data [2, 63]. Besides, in-distribution clean samples [45], adversarial samples near the decision boundary [30] or images with an imperceptible logo embedded by an encoder [37] can also be exploited as watermark data. In terms of the target class setting, existing black-box DNN watermarks can be categorized into fixed-class watermarks, where all the watermark data are paired with the identical target class (such as [27, 35, 63]), and non-fixed-class watermarks, where each watermark data is paired with its own target class (such as [2, 20, 30, 37, 45]). These complex designs pose additional challenges to a practical attacker, who usually has no knowledge of the underlying watermark schemes.

2.2 Watermark Removal Attacks

Black-box model watermarks are to some extent similar to backdoor attacks, as they both establish connections between some specified data and the target labels. Therefore, classical methods in backdoor defense, such as pruning, finetuning or trigger reverse-engineering, are usually considered potential threats for black-box watermarks [2, 27, 63]. Recently, some attacks especially targeting black-box watermarks are also proposed.

Existing watermark removal attacks can be mainly categorized into three types as follows, with the pros-and-cons of attacks from each category summarized in Table 1.

- **Pruning-based Attacks**: They prune the redundant weights [54] or neurons [38] in DNNs. However, to invalidate the underlying watermarks, these methods need to prune a large proportion of weights or neurons, which causes an unacceptable utility loss.
- **Finetuning-based Attacks**: They continue to train the target model for a few epochs with some carefully designed finetuning techniques, such as learning rate schedule [10, 11], dataset augmentation [39], weight regularization [47], or continue learning with attention distraction [65]. Unfortunately, these attacks could only breach a subset of watermarks. Furthermore, due to stringent training restrictions, they require a large amount of clean data to maintain utility.
- **Unlearning-based Attacks**: They are mainly designed for black-box watermarks whose watermark data have the fixed pattern and the identical target class [3, 49]. For instance, the *Laundering* attack [3] uses the classical trigger reverse-engineering technique in backdoor defense literature *NeuralCleanse* [56], followed by an unlearning process. However, these attacks pose strong assumptions on the forms of underlying watermarks and are therefore not applicable to physical scenarios where attackers have no knowledge of the target watermarks. Also, current unlearning-based attacks all need access to source training data, which further limits their applicable scenarios.

Note that there also exists another type of attack aiming at training a new surrogate model using the knowledge transferred from the given watermarked model, i.e., **model extraction attacks** [27, 42, 60]. However, extraction attacks typically require a large query dataset and entail heavy computation costs [47]. Therefore, we do not include these attacks in the main experiment, but will make a comparison with them under different data settings in Section 7.

3 SECURITY SETTINGS

3.1 Mechanism of Black-box Watermarking

The true model owner is denoted as O. During the training, O embeds a watermark into the target model by training on both the clean dataset $X = \{(x_i, y_i)\}_{i=1}^{N_1}$ $(y_i \in \{0, 1, \ldots, C-1\})$ from its main task, where C is the total number of the classification classes, and on a set of specially-crafted watermark data $X_w = \{(x_{wi}, y_{wi})\}_{i=1}^{N_2}$ $(y_{wi} \in \{0, 1, \ldots, C-1\})$, where target classes y_{wi} can be either identical or sample-specific. We denote the watermarked model as f_w. In face of potential copyright infringement, O hopes to verify the model ownership by comparing the suspect model's predictions on X_w with the pre-defined target labels, via the provided API (i.e., black-box access), as shown in Figure 1.

By inspecting the official watermark implementations, we observe the following technical designs which the owner O will take to meet the three requirements in Section 2.1. Specifically, ❶ to satisfy the **fidelity** purpose, O usually choose an over-parameterized model with enough capacity, which is trained on both normal samples and watermark samples, with normal ones being dominant in quantity [2, 37], i.e., $N_2 \ll N_1$. ❷ To ensure the watermark **integrity**, O typically crafts watermark samples (x_{w_i}, y_{w_i}) that are significantly different from the normal samples in the target class y_{w_i}, whose size N_2 cannot be too small as well, to claim the ownership with high confidence [27, 35, 42]. ❸ To enhance the **robustness** of the embedded watermark, O tends to intentionally increase the involvement of the watermark data in the training

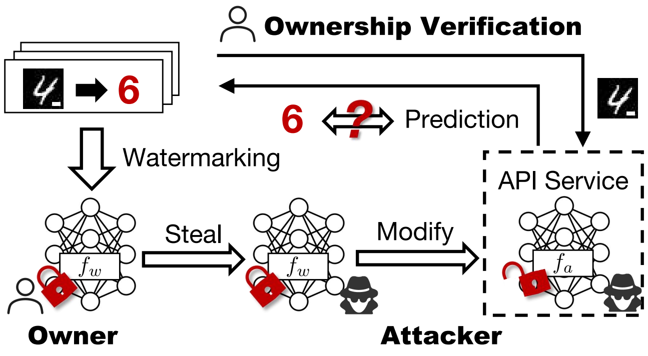

Figure 1: An illustration of our threat model. The model owner trains a watermarked model f_w with a secret set of watermark samples, and hopes to perform ownership verification via API queries for suspect models. On the other hand, the attacker has managed to acquire the white-box f_w, and aims to derive a surrogate model f_a with the watermark removed, evading subsequent ownership verification.

process. For instance, both Adi et al. [2] and Lederer et al. [33] concatenate the independently sampled normal data and watermark data into a data batch during each training iteration. Jia et al. [27] intentionally train the model on watermark samples and on normal samples alternatively. Lukas et al. [42] explicitly boost the ratio of watermark data when constructing the total training set [41].

3.2 Threat Model

Attack Scenario. In Figure 1, the attacker $\mathcal{A}$ is unwilling to train a model of their own, due to the lack of training data, computation resource or expertise. Instead, $\mathcal{A}$ attempts to infringe on the intellectual property of f_w trained by O, and then host a similar service illegally or abuse it to make a profit, while evading the subsequent watermark verification process.

Similar to the settings in many watermark robustness studies [3, 10, 34, 42, 65], we assume the attacker has managed to acquire the white-box access to the target model f_w, i.e., they can observe the model parameters as well as modify them. This access could be achieved by, e.g., directly downloading the model after dishonestly agreeing with the official license (on open-source model hubs such as Hugging Face [17], DNNs might be released under licenses permitting research but prohibiting commercial use [14]), decrypting the possibly encrypted model from memory (DNNs on end devices) [50], or stealing functional replicas of DNNs via API access (MLaaS) [53]. Additionally, we assume that $\mathcal{A}$ is aware of the existence of the watermark in f_w, and therefore their goal is to derive a surrogate model f_a from f_w with similar performance while invalidating the ownership verification.

Attack Budget. Specifically, as shown in Table 1, ❶ $\mathcal{A}$ has no knowledge of the specific watermark algorithm or the watermark data X_w that O used (i.e., watermark-agnostic). ❷ $\mathcal{A}$ hopes to significantly lower f_a's watermark verification accuracy on X_w, ❸ while preserving its clean accuracy on the main task. To make this attack practical, ❹ $\mathcal{A}$ also wants to relax the dataset access requirement as much as possible.

Previous removal attacks may require a substantial subset of the clean training data X [2, 10, 63], or a considerable size of unlabeled data collected from open sources as proxy data [21, 47]. However, they usually ignore the difficulty of collecting high-quality data at such a scale. In this work, we mainly consider the attacker with limited data access, as in [3, 65]. The following three assumptions on dataset availability characterize attackers from being limited in knowledge to almost of zero knowledge:

- **In-distribution**: A limited number of correctly labeled samples (either manually by humans, or automatically by the well-training watermarked model) are available from the same distribution of the target model's main task. This setting is realistic since these in-distribution data are often hard to acquire.
- **Transfer**: A small amount of unlabeled data from a transfer distribution is available. Note that the distribution of these unlabeled data also matters, since finetuning the target model on samples from a completely different distribution might lead to catastrophic forgetting [28].
- **Data-free**: No additional data is available. This setting further relaxes the dataset access requirement for the attacker. To the best of our knowledge, our work is the first to formally consider data-free black-box watermark removal settings.

4 GENERAL FRAMEWORK OF DEHYDRA

In general, our Dehydra adopts the following two-stage attack pipeline: **(1) Watermark Recovering:** We first leverage model inversion technique to recover some samples close to real watermark data (§4.1). **(2) Watermark Unlearning:** We next deliberately unlearn these samples during the finetuning process, along with the auxiliary dataset $\mathcal{A}$ possesses (§4.2). The procedures bear a resemblance to dehydration reactions in chemistry, where water molecules are expelled from compounds, inducing the formation of new substances. An overview of our general attack framework is shown in the upper region of Figure 2.

4.1 Watermark Recovering

Despite the complexity of various designs of watermarks, including different watermark patterns and strategies for setting the target classes, we summarize their commonality that, to implement robustness, existing watermarks exploit the over-parameterization property of DNNs to intentionally memorize the watermark data correlated to the pre-defined target labels. Note that training samples with higher repetition and memorization levels may face greater reconstruction risks [7]. Inspired by this, we believe the model internals should be a sufficient source to reconstruct the watermark information, and hence attempt to use model inversion techniques to reverse-engineer samples close to the real watermark data from a watermarked model.

Concretely, we use optimization-based model inversion to recover the samples hidden in the watermarked model f_w. Given a target class c and a batch of randomly initialized samples $B = \{\hat{x}_i\}_{i=0}^{M-1}$ ($\hat{x}_i$ keeps the same shape as the images of f_w's main task, while M is the number of samples to be optimized for the c-th class, we try to reverse-engineer samples which are similar to the watermark

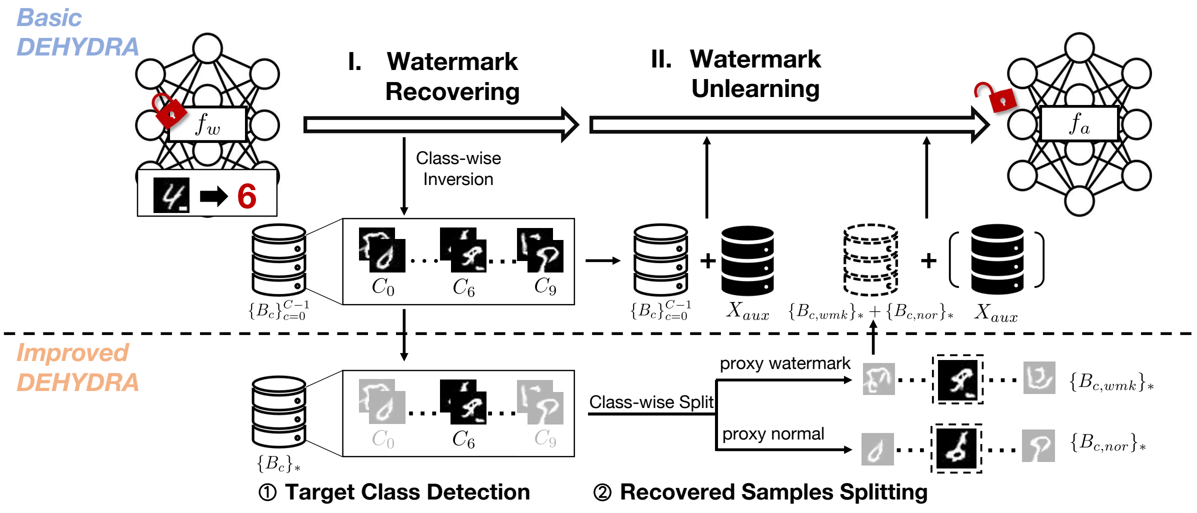

Figure 2: The overview of our DEHYDRA. The upper region shows our basic attack, comprising two stages, watermark recovering (§4.1), which reconstructs batch samples $\{B_c\}_{c=0}^{C-1}$ close to real watermark data from each class, and watermark unlearning (§4.2), which unlearns the recovered samples during finetuning, along with the auxiliary dataset X_{aux}. The lower region shows our improved designs, target class detection (§5.1) and recovered samples splitting (§5.2). The former detects the target classes after recovering (e.g., C_6 in this case), and only retains batches $\{B_c\}_*$ from those classes. The latter performs class-wise splitting on each B_c before unlearning, to identify samples closer to watermark or normal data, i.e., $\{B_{c,wmk}\}_*$ and $\{B_{c,nor}\}_*$, respectively.

data correlated to the target class c by optimizing

$$\min_B\Big(\sum_{\hat{x}_i\in B}\mathcal{L}(f_w(\hat{x}_i),c)\Big)+\mathcal{R}(B), \tag{1}$$

where $\mathcal{L}(\cdot)$ is the cross-entropy loss, and the regularizer $\mathcal{R}(\cdot)$ poses our prior knowledge on the watermark data.

In addition to classic priors on natural images (e.g., ℓ_2 regularization and total variation regularization [44]), our recovering algorithm mainly exploits the information stored in the batch normalization (BN) layers [26], as is done in [61, 62]. BN layers are widely adopted in mainstream DNNs [23, 24, 51], and keep useful statistics of the training data, capturing both low-level and high-level features in the neural network. Since the owner O tends to intentionally increase the involvement of the watermark data during the training, the watermark data X_w are expected to play an important role in the BN statistics of the target model f_w.

To better utilize the watermark information hidden in the BN layers, we employ another feature regularization term $\mathcal{R}_{bn}(\cdot)$ to guide the recovered samples closer to the real watermark data:

$$\mathcal{R}_{bn}(B)=\sum_{l=1}^{L-2}\|\mu_l(B)-\mu_l\|_2^2+\|\sigma_l^2(B)-\sigma_l^2\|_2^2, \tag{2}$$

where L is the total number of BN layers in f_w, μ_l and σ_l^2 are the running batch-wise mean and variance stored in the l-th BN layer, and $\mu_l(\cdot)$ and $\sigma_l^2(\cdot)$ are current statistics of feature maps before the l-th BN layer. According to [25, 48], the last several layers of a DNN capture high-level features, which are more correlated with the model's prediction behavior. To encourage the exploration in class c's decision space and capture more watermark information, we intentionally dismiss the guidance of the last two BN layers.

In summary, the regularization term in Equation 1 is finally:

$$\mathcal{R}(B)=\alpha_{\ell_2}\mathcal{R}_{\ell_2}(B)+\alpha_{tv}\mathcal{R}_{tv}(B)+\alpha_{bn}\mathcal{R}_{bn}(B), \tag{3}$$

where $\mathcal{R}_{\ell_2}(B)$ and $\mathcal{R}_{tv}(B)$ penalize the ℓ_2 norm and the total variation of the recovered watermark data. The detailed forms can be found in Appendix A.1.

Since the attacker $\mathcal{A}$ has no knowledge of the underlying target classes, $\mathcal{A}$ has to conservatively repeat the optimization step in Equation 1 for all possible classes and finally obtain C batches of recovered samples $\{B_c\}_{c=0}^{C-1}$, where B_c indicates the inverted batch of samples towards class c. These recovered samples are expected to contain enough watermark information of f_w.

Note that although the BN layers keep the statistics of the total training set, including samples of all classes, our watermark recovering method is actually using an aggressive class-wise inversion scheme. This is because our goal is to *recover samples close to the watermark data as much as possible here, instead of to generate realistic images*. Since the watermark data may constitute a large contribution to the BN statistics during the training, we propose this scheme to greedily absorb information in BN layers, preventing it from diffusing into other classes. This design is especially effective against watermarks with fixed target classes, compared to the arbitrary-class scheme in conventional inversion methods [61]. We validate this point in Section 6.4.2.

4.2 Watermark Unlearning

Directly unlearning these recovered samples $\{B_c\}_{c=0}^{C-1}$ without other constraints may cause catastrophic forgetting, bringing irreversible significant damage to the normal performance. In the current framework, we consider two data settings for $\mathcal{A}$'s auxiliary data X_{aux}, in-distribution and transfer. This basic framework is extended to the data-free setting with more improved designs in Section 5.

Assuming the attacker $\mathcal{A}$ has some in-distribution data $X_{aux} = \{(x_i, y_i)\}$ from f_w's main task, they can leverage these labeled samples to preserve model's performance during unlearning. When $\mathcal{A}$ only has some unlabeled data $\{x_i\}$ from a different distribution, similar goals can be achieved by leveraging model f_w as an oracle to generate pseudo-labels for each sample, yielding the transfer dataset $X_{aux} = \{(x_i, f_w(x_i))\}$.

Starting from f_w, the attacker can finetune the surrogate model f_a under the following optimization objective:

$$\min_{\theta_a} \underbrace{\sum_{(x_i,y_i)\in X_{aux}} \mathcal{L}(f_a(x_i), y_i)}_{\text{preserving original performance}} + \underbrace{\alpha_{KL} \sum_{B_c\in\{B_c\}_{c=0}^{C-1}} \sum_{\hat{x}_i\in B_c} KL(f_a(\hat{x}_i), y_{soft})}_{\text{removing watermark}}, \tag{4}$$

where θ_a is the parameters of f_a, $KL(\cdot)$ is the Kullback–Leibler divergence, y_{soft} is a soft unlearning target of length C, where each element equal to $1/C$. Note that we are using a more consistent unlearning target, compared to the hard random labels, which might lead to gradient conflicts and cancellations during fine-tuning.

The first term in Equation 4 stabilizes the general prediction behavior of f_a and thus preserves its performance, while the second term encourages f_a to unlearn the watermark-related information. In this way, we successfully derive a surrogate model f_a to prevent successful watermark verification.

5 IMPROVED DESIGNS FOR DEHYDRA

In this section, we enhance the general framework of Dehydra in the following directions.

(1) **Improved Recovering via Target Class Detection**: First, existing black-box watermarks consist of those with fixed target classes and with non-fixed target classes. For watermarks with fixed target classes (such as *Content* [63] and *EWE* [27]), recovering and unlearning watermark data from other classes bring no benefit to watermark removal, and can even sacrifice the utility of the surrogate model. Therefore, after the original watermark recovering stage, we further distinguish whether the underlying watermark has a fixed target class, and detect its target class if so. We only retain recovered samples belonging to the target class for a fixed-class watermark (§5.1).

(2) **Improved Unlearning via Recovered Sample Splitting**: Second, the inverted samples derived by Equation 1 may also contain important information of the main task. Indiscriminately unlearning all the inverted samples from the target classes might also hurt the model utility. Therefore, before the original watermark unlearning stage, we split out samples closer to the real watermark data from the recovered ones and improve the finetuning loss accordingly (§5.2).

Figure 2 shows two improved designs and the complete workflow. The two improvements mainly exploit the recovered samples instead of the auxiliary data, enabling our attack to be extended to a data-free setting for watermarks with a fixed target class.

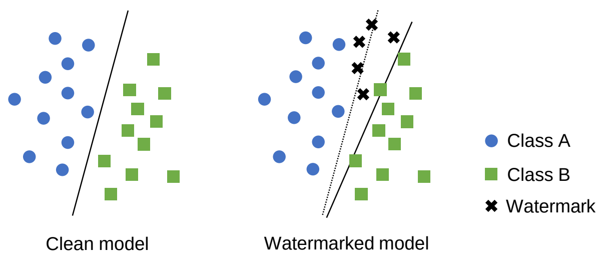

Figure 3: An illustration of the watermark smoothness discrepancy hypothesis. Left: A clean model with its decision boundary. Right: A fixed-class watermark is embedded, with all watermark samples labeled to Class A.

5.1 Target Class Detection

• **Insight: Watermark Smoothness Discrepancy.** By inspecting existing DNN watermarking implementations [2, 27, 33, 42, 63], we find that for a watermark with a fixed target class, the model tends to learn much more diverse data correlated to the target class during training, compared with other labels, due to the additional incorporation of special watermark samples (for the integrity purpose). Moreover, the model also learns samples belonging to the target class more prominently, because of the higher participation ratio of these watermark data (for the robustness purpose).

Based on these observations, we propose the *watermark smoothness discrepancy* hypothesis that, for a DNN model embedded with a fixed-class watermark, the loss landscape (i.e., the structure of loss values around the parameter space) is smoother with respect to samples of the target label, compared to those of other labels. The smoothness property is mathematically defined via the Hessian matrix of loss function [18, 36], which is usually computationally expensive. We instead approximate the smoothness using the performance of model under perturbations. A conceptual illustration is shown in Figure 3, where compared with the clean model, the watermarked model demonstrates higher smoothness on Class A and lower on Class B, due to the additional incorporation of watermark samples and the increased involvement during training.

• **Theoretical Justification.** To provide theoretical evidence, we consider training a watermarked binary classification linear model $f(x) = \text{sign}(\langle w, x\rangle + b)$ on a dataset following mixture Gaussian distributions, together with another set of fixed-class watermark data. The detailed problem definition, theorems and proofs are all deferred to Appendix D due to the page limit.

Our conclusions are (Remark on Theorem D.1): *For a fixed-class watermark, the target class will exhibit a higher smoothness (both input-level and parameter-level) compared to other classes if the watermark samples are (1) diverse internally and different from the normal ones in the target class, and (2) sampled much more frequently. During proof, we also find the additive input-level and parameter-level noise would further enlarge the class-wise discrepancies.*

In existing black-box watermarking implementations, the designs of watermark data for the integrity purpose correspond to the first condition, while the training techniques for the robustness purpose satisfy the second condition. Therefore, we deduce that the watermark smoothness discrepancy hypothesis should also hold for realistic watermarked DNNs.

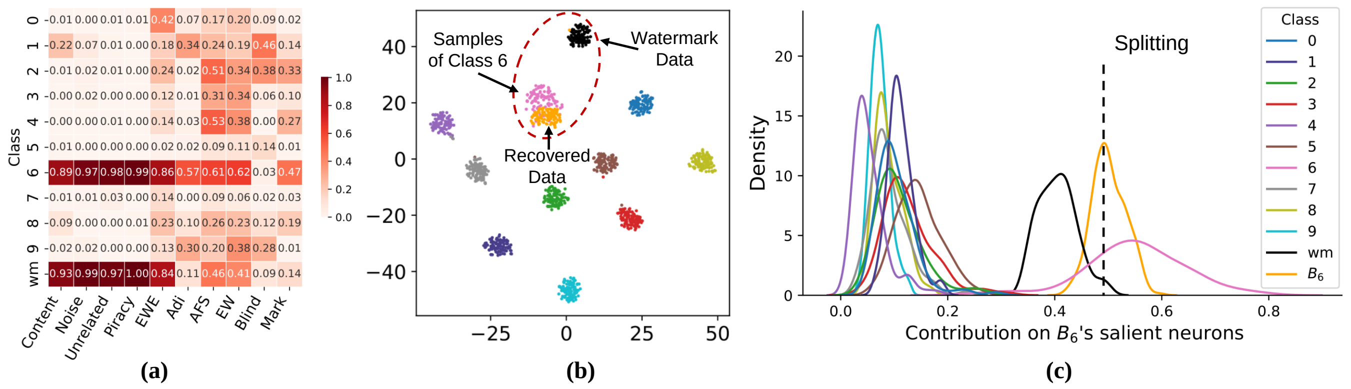

Figure 4: Pilot study results of the improved designs. (a) Class-wise smoothness analysis of the five fixed-class and five non-fixed-class watermarks on CIFAR-10. (b) Activations visualization of a model protected by the *Content* watermark with target class 6. (c) Distribution of sample contributions per class, on the salient neurons of the recovered samples B_6.

Note that another line of work focuses on class-wise robustness discrepancies [6, 52] and empirically discovers that, for a normal model trained on a balanced dataset, classes with smaller *inter-class* semantic distances are more vulnerable. Our findings are exactly complementary to theirs: the classes with a larger *intra-class* variance and sampling ratio should be more robust.

• **A Pilot Study.** To provide a more comprehensive investigation, we empirically evaluate the smoothness on ten realistic models protected by different black-box watermarks. Following the previous interpretations of smoothness, we define a metric called *SmoothAcc*. Given a set of samples X, we calculate the average prediction accuracy of $(x_i, y_i) \in X$ under perturbations:

$$SmoothAcc(X) = \frac{1}{|X|} \sum_{(x_i,y_i)\in X} \mathbb{P}(\operatorname*{argmax}_j f_w(x_i+\epsilon_1;\theta_w+\epsilon_2)_j = y_i), \tag{5}$$

where $\epsilon_1 \in \mathcal{N}(0, \sigma_1^2 I_1)$ is the input-level Gaussian noise and $\epsilon_2 \in \mathcal{N}(0, \sigma_2^2 I_2)$ is the parameter-level Gaussian noise.

We construct ten watermarked ResNet-18 [23] models on CIFAR-10 [29] dataset, embedded with five fixed-class watermarks (with the target class set to 6[1]) and five non-fixed-class watermarks respectively (more details are described in Section 6.1). For each model, to comprehensively evaluate its smoothness on various data categories, we collect 11 batches of samples, including ten batches of normal samples, each from a class in CIFAR-10, and one batch from real watermark samples, with a batch size of 100. During smoothness analysis, we set the standard deviations of the Gaussian noise $\sigma_1 = 0.5$ and $\sigma_2 = 0.015$[2], and repeatedly sample the Gaussian noise 100 times for each sample to approximate the possibility $\mathbb{P}(\cdot)$ in Equation 5. The results are shown in Figure 4(a).

Evidently, for fixed-class watermarks, the models exhibit much higher smoothness on samples from the target class, including normal samples and watermark samples. This class-wise difference is less evident for non-fixed-class watermarks, where models achieve moderate smoothness on several classes simultaneously. These results strongly support our hypothesis.

• **Technical Designs.** Based on the observations above, we assume that the watermark smoothness discrepancy phenomenon also exists, when the target model f_w is embedded with a fixed-class watermark, in the class-wise inverted samples $\{B_c\}_{c=0}^{C-1}$. Supposing the fixed target label is y_w, then the class-wise smoothness difference would also be significant in $\{B_c\}_{c=0}^{C-1}$ as the inverted batch B_{y_w} from the target label will cover the space of both normal samples of y_w and watermark samples.

In light of this, we propose to evaluate the smoothness of f_w on $\{B_c\}_{c=0}^{C-1}$ after the watermark recovering, i.e., to calculate $SmoothAcc(B_c)$ for $c \in \{0, 1, \ldots, C-1\}$ respectively. Each $\hat{x}_i \in B_c$ is temporarily labeled as class c during the smoothness evaluation. Next, we sort the batches $\{B_c\}_{c=0}^{C-1}$ according to their $SmoothAcc(B_c)$ values in the descending order, yielding sorted $\{B_{s_0}, B_{s_1}, \ldots, B_{s_{C-1}}\}$. We detect whether f_w is embedded with a fixed-class watermark, and, if so, follow the criterion below to determine the target class: **a)** If the gap between the recovered two batches evaluated with the highest smoothness, i.e., $SmoothAcc(B_{s_0}) - SmoothAcc(B_{s_1})$, is larger than a certain threshold T, we regard the underlying watermark as a fixed-class watermark, and take the target label as s_0. We then only retain recovered samples B_{s_0} and drop other batches. **b)** Otherwise, we regard the underlying watermark as a non-fixed-class watermark, and keep all the recovered batches $\{B_c\}_{c=0}^{C-1}$ since each class c might have several correlated watermark samples. We additionally record labels of the last two batches, s_{C-1} and s_{C-2}, as the least-likely label and second-least-likely label respectively.

In the remainder of this paper, we use $\{B_c\}_*$ to denote $\{B_{s_0}\}$ if a fixed-class watermark is detected, and $\{B_c\}_{c=0}^{C-1}$ otherwise. Consequently, after the watermark recovering and target class detection, we get batch(es) samples $\{B_c\}_*$ close to real watermark data X_w.

5.2 Recovered Samples Splitting

• **Insight: Normal Data Dominance.** We have previously explained the necessity of the class-wise inversion scheme for watermark recovering (§4.1), but we find that this scheme would induce

[1]The target classes of the five investigated fixed-class watermarks are set to 6 arbitrarily in this pilot study. We use a randomly chosen label under ten repetitive tests for a systematic study in Section 6.3.

[2]We at first empirically set the standard deviations of the input-level and parameter-level Gaussian noise. Subsequently, we independently increment each standard deviation until the smoothness discrepancy is evident.

competitive objectives during recovering. Recall the two key components of the recovering objective in Equation 1, i.e. the targeted classification loss $\sum_{\hat{x}_i \in B} \mathcal{L}(f_w(\hat{x}_i), c)$ and the BN regularization term $\sum_{l=1}^{L} \|\mu_l(B) - \mu_l\|_2^2 + \|\sigma_l^2(B) - \sigma_l^2\|_2^2$. For the recovering process on target class c, the targeted classification loss will lead all the optimized samples B_c consistently classified to c, while the BN regularization term will guide the neuron activations of B_c statistically close to the activations average of the total training set. Considering that the normal samples constitute the main distribution, different from the watermark distribution, the recovered B_c should be closer to class c's clean data (which belongs to the main distribution) than the watermark samples labeled to c.

Based on the above analysis, we derive another hypothesis named *normal data dominance*, which is intrinsically connected with our class-wise inversion scheme. Specifically, for a watermarked model (with either fixed or non-fixed target classes), the class-wise recovered samples are expected to be closer to normal samples compared with the watermark samples in that class.

• **A Pilot Study.** To gain insights, we analyze the neuron activations of the model protected by *Content* [63] watermark here with fixed target class 6. We first collect 12 batches, including ten batches of normal samples from each class, one batch of watermark samples and one batch of recovered samples B_6 from target class 6, each with a batch size 100. We feed each batch into the watermarked model, extract the activations of the penultimate layer and perform t-SNE dimensionality reduction [55]. As shown in Figure 4(b), the recovered samples (the orange cluster) lie slightly close to the global activations center, and hence closer to the normal samples of class 6 (the pink cluster) than the watermark samples (the black cluster, lying separately from the main distribution). This is because, the watermark task is usually irrelevant to the model's primary task, making it extremely challenging to recover high-quality watermark samples. Although we have employed the class-wise BN regularization term to enhance the coverage of the recovered B_6 over the real watermark samples (§4.1), it would also inevitably guide the recovered samples towards the activation average of the total training set, conforming to our analysis above.

Next, we delve into the composition of the activations of the recovered samples B_6. Since B_6 are pulled statistically close to the activation center of training data of all classes, it is expected that samples of B_6 will inevitably activate redundant neurons of class c or neurons important to other classes. To prevent the noise from these irrelevant activations, we focus on B_6's salient neurons. Concretely, we utilize the *importance score* for recognizing the salient neurons [12]. Given a batch of samples X_c of class c, the score extracts the neuron activations at the l-th layer of model f_w, i.e., $f_{w(l)}(X_c)$, and calculates an importance score for each j-th neuron: $Impt(f_{w(l)}(X_c)_j) = \frac{\mu_j}{\sigma_j}$, where μ_j and σ_j are the mean and the standard deviation of the j-th neuron estimated over X_c. We refer to the neurons with top-5% importance score as salient neurons of X_c, denoted as $SN(X_c)$.

Next, we capture the salient neurons of the recovered samples B_6 and explore the functionality of these consistently activated neurons. For each sample $x_i \in X_c$ among the 12 batches collected above, we define its total activation values on the salient neurons of B_6 as its contribution:

$$Contribution(x_i) = \sum_{j \in SN(B_6)} f_{w(l)}(x_i)_j. \tag{6}$$

Then we plot the distribution of sample contributions of each class, distinguished in color. In Figure 4(c), for samples belonging to class 6, normal samples contribute the most, followed by samples in B_6 itself, and finally the watermark samples. This indicates that $SN(B_6)$ are more consistently activated by normal samples of class 6 than the watermark samples, which validates our hypothesis.

More empirical evidence is provided in Figure 8 of Appendix C.1, where, in most cases, $SN(B_6)$ exhibit the highest correlation with the normal samples of class 6. It is worth noting, for non-fixed-class watermarks, $SN(B_6)$ also has a high correlation with the watermark samples besides the normal samples, possibly due to the more complex input-output pairing relations, leaving the watermark data more entangled with the normal data.

• **Technical Designs.** As the class-wise recovered samples B_c might also contain important information of the normal samples in class c, in this section, we exploit the hypothesis of normal data dominance to split B_c into two parts, namely proxy normal data and proxy watermark data. We develop a neuron-level criterion: For each class c, we extract the salient neurons $SN(B_c)$, and take the subset achieving the largest contributions as proxy normal data, leaving the rest as proxy watermark data. Algorithm 1 in Appendix A.2 shows the detailed procedures.

Next, we formally discuss the underlying rationale. An illustrative example is shown in Figure 4(c), where we split the recovered samples B_6 (the orange distribution) according to their contributions on $SN(B_6)$. After splitting, the right part of B_6 are taken as proxy clean data, which are close to the normal samples of class 6 and preserve the normal functionalities specific to class 6. The left part of B_6 are proxy watermark data, either correlating to the real watermark data, or noisily activating other classes' salient neurons (which are inessential for normal performance).

Given the recovered samples $\{B_c\}_*$ obtained via watermark recovering and target class detection, the algorithm will perform class-wise splitting on each batch B_c, returning proxy normal data $\{B_{c,nor}\}_*$ and proxy watermark data $\{B_{c,wmk}\}_*$. Finally, we improve the finetuning loss accordingly.

If the target model is detected with a fixed-class watermark (i.e., the target class s_0), we only need to address samples from class s_0, denoted as $\{B_{c,wmk}\}_* = \{B_{s_0,wmk}\}$ and $\{B_{c,nor}\}_* = \{B_{s_0,nor}\}$. In this way, the attacker can finetune the surrogate model f_a under the following objective:

$$\min_{\theta_a} \underbrace{\left(\sum_{(x_i,y_i) \in X_{aux}} \mathcal{L}(f_a(x_i), y_i) \right) + \sum_{\hat{x}_i \in B_{s_0,nor}} \mathcal{L}(f_a(\hat{x}_i), s_0)}_{\text{preserving original performance}} + \underbrace{\sum_{\hat{x}_i \in B_{s_0,wmk}} \mathcal{L}(f_a(\hat{x}_i), \text{rand}_{s_0})}_{\text{removing watermark}}, \tag{7}$$

where rand_{s_0} denotes a random label in $\{0, 1, \ldots, C-1\}$ except s_0 itself. This unlearning target is stronger than minimizing $KL(\cdot, y_{soft})$

in Equation 4, due to the enhanced specificity brought by target class detection, and also better than maximizing $\mathcal{L}(\cdot, s_0)$, due to the easier convergence. Note that in the objective function above, we additionally exploit the proxy normal data $B_{s_0,nor}$ to preserve f_a's original performance. This design enables our Dehydra to be extended to a data-free setting: when the auxiliary data is not available (i.e., the first term enclosed by $(\cdot)$ in the above equation is optional), we can still perform watermark unlearning only using the recovered samples.

For models detected with a non-fixed-class watermark, $\{B_{c,wmk}\}_*$ contains C batches of proxy watermark data. Then $\mathcal{A}$ can finetune f_a under this objective:

$$\min_{\theta_a} \underbrace{\sum_{(x_i,y_i)\in X_{aux}} \mathcal{L}(f_a(x_i), y_i)}_{\text{preserving original performance}} + \underbrace{\sum_{B_{c,wmk}\in\{B_{c,wmk}\}_*} \sum_{\hat{x}_i\in B_{c,wmk}} \mathcal{L}(f_a(\hat{x}_i), y_\star)}_{\text{removing watermark}}, \tag{8}$$

where $y_\star$ denotes the least-likely label s_{C-1} (obtained during the target class detection process) for $B_{c,wmk} \in \{B_{c,wmk}\}_*$ and $c \neq s_{C-1}$, while the second-least-likely label s_{C-2} when $c = s_{C-1}$. This proxy label unlearning target is also stronger than that in Equation 4 due to the enhanced specificity, while maintaining the consistent descending gradients across the labels. Different from the design for fixed-class watermarks, we do not use proxy normal data to further preserve the model performance. This is because the normal data dominance phenomenon is less evident for non-fixed-class watermarks (Figure 8), and thus the salient neurons of the split proxy normal data might still have some intersection with those of the real watermark data. Therefore, leveraging these proxy normal data for preserving utility might hinder the watermark removal.

6 EXPERIMENTS

6.1 Overview of Evaluation

To evaluate the performance of Dehydra, we perform a comprehensive study on ten mainstream black-box watermarking schemes, under various benchmark datasets, DNN architectures and data availability settings. Before presenting the detailed evaluation results, we first provide a concise introduction to the experimental setups.

• **Datasets and Victim Models.** Following the settings in [3, 27, 34, 47], we construct victim models on three benchmark datasets, namely, MNIST [32], CIFAR-10 and CIFAR-100 [29]. The respective DNN architectures are LeNet-5 [31], ResNet-18 [23] and ResNet-34. We embed black-box watermarks into the victim models during the training. Then we perform watermark removal attacks, including Dehydra and baseline attacks, to evaluate their effectiveness.

• **Target Watermark Schemes.** Our evaluation covers ten mainstream black-box DNN watermarking schemes published at top-tier conferences (some of which are developed by industry leaders such as IBM): *Content*, *Noise*, *Unrelated* [63], *Piracy* [35], *EWE* [27], *Adi* [2], *AFS* [30], *EW* [45], *Blind* [37], *Mark* [20]. These schemes feature diverse designs in the watermark data and target label settings, and are mostly evaluated in watermark robustness surveys [33, 34, 42]. For more backgrounds, please refer to Appendix B.1.

During implementation, we strictly followed the specifications and hyperparameters in the original papers (with the target class set to 6 for those fixed-class watermarks) to prepare the watermarked models. We also referred to some recent open-source replications [33, 42] to ensure our implementation is faithful. For more implementation details and the performance of the watermarked models, please refer to Appendix B.2 and B.3.

• **Baseline Removal Attacks.** We compare our Dehydra with the following three types of removal attacks, comprising six baseline attacks in total.

- *Pruning-based attacks:* **(1)** ***Pruning*** **[54, 63]** directly sets a proportion of DNN parameters with the smallest absolute values to zero. **(2)** ***Fine-pruning*** **[38]** prunes neurons that are infrequently activated by normal data, followed by a finetuning process.
- *Finetuning-based attacks:* **(3)** ***Finetuning*** **[10, 11]** specifically finetunes the target model using a large learning rate, together with a carefully-designed scheduler. **(4)** ***Regularization*** **[42, 47]** finetunes the target model with a large L2 regularization on the parameters. **(5)** ***Distraction*** **[65]** finetunes the target model on in-distribution or transfer data, together with another set of lure data that distracts the model's attention away from the watermark.
- *Unlearning-based attacks:* **(6)** ***Laundering*** **[3]** leverages trigger reverse-engineering methods [56] in backdoor defense literature to recover watermark data, followed by neuron resetting and model retraining, to remove backdoor-based watermarks.

The implementation details of these baseline attacks are clarified in Appendix B.4.

• **Dataset Availability Settings.** We mainly focus on the realistic settings where the attacker has limited data access, including three different scenarios as follows: **(1) In-distribution setting.** The attacker has 1000 correctly-labeled normal samples (2% of the size of the CIFAR-10 training set), following [65]. **(2) Transfer setting.** Here the attacker cannot access the source training data, but is assumed to have 2000 unlabeled samples from another distribution (e.g., CIFAR-100 for target models trained on CIFAR-10, following [21, 65]). **(3) Data-free setting.** In this setting, the attacker cannot access any samples.

• **Implementation of Dehydra.** During watermark recovering, we set hyper-parameters $M = 250, \alpha_{\ell_2} = 0.01, \alpha_{tv} = 0.03, \alpha_{bn} = 0.1$. We use $tanh(\cdot)$ to constrain the batch data within a valid range, and optimize the recovering objective with the Adam optimizer of learning rate 0.1. For target class detection, we set $\sigma_1 = 1.0, \sigma_2 = 0.03$ for MNIST and $\sigma_1 = 0.5, \sigma_2 = 0.015$ for CIFAR-10 and CIFAR-100 respectively. The detection threshold T is conservatively set to 0.4 for MNIST and CIFAR-10, while 0.3 for CIFAR-100. For splitting the recovered samples, we set l to be the penultimate layer of the target model, and saliency ratio $\beta = 0.95$, split ratio $\gamma = 0.5$ in Alg.1. During watermark unlearning, we set $\alpha_{KL} = 15$ to ensure the unlearning strength for the basic Dehydra and the uniform loss weights for the improved Dehydra. The model is finally finetuned for 10 epochs using the SGD optimizer with the batch size 128. The learning rate is set to 0.01 when auxiliary data is available

Table 2: Comparison of removal attacks against black-box DNN watermarks under the in-distribution setting on CIFAR-10. x / y denotes the clean accuracy / rescaled watermark accuracy, and values behind ± report their standard deviations respectively in 5 repetitive tests. The rescaled watermark accuracy values below 50% are bolded and underlined. The left five columns are attack results of the fixed-class watermarks, while the right five are of the non-fixed-class watermarks.

Attacks	***Content***	***Noise***	***Unrelated***	***Piracy***	***EWE***	***Adi***	***AFS***	***EW***	***Blind***	***Mark***
None	93.8 / 100.0	94.1 / 100.0	93.7 / 100.0	93.3 / 100.0	94.0 / 100.0	93.9 / 100.0	93.0 / 100.0	92.9 / 100.0	91.2 / 100.0	94.0 / 100.0
Fine-pruning	86.4 / **44.7**	87.4 / 67.0	85.9 / 52.6	69.5 / **37.8**	88.9 / 70.2	85.4 / 59.8	87.3 / 80.4	87.6 / 83.3	77.0 / 53.8	87.3 / 61.1
Finetuning	85.5 / **46.4**	85.1 / 84.9	84.6 / **47.1**	66.1 / **40.9**	86.9 / 81.2	85.0 / 56.3	90.4 / 99.4	90.1 / 98.8	79.9 / 55.0	85.8 / 57.1
Regularization	60.3 / 100.0	54.7 / 91.5	82.8 / 100.0	76.9 / 100.0	68.2 / **47.6**	55.4 / 73.8	72.3 / 81.6	69.0 / 82.1	69.8 / 83.6	75.7 / 87.2
Distraction	87.4 / **47.5**	89.5 / 98.1	88.5 / 51.5	70.6 / **40.9**	85.8 / 52.4	89.4 / 52.2	92.5 / 99.4	92.5 / 100.0	89.8 / 91.2	86.0 / 51.3
Laundering	91.1 / **48.1**	90.7 / 77.4	88.8 / 100.0	83.8 / 69.9	91.1 / 70.2	87.0 / 69.7	89.3 / 100.0	89.3 / 98.8	86.3 / 73.1	91.1 / 94.8
Dehydra$_{\text{Basic}}$	88.8 / **44.7**	88.1 / **5.7**	88.1 / **44.9**	83.3 / **44.0**	89.0 / **48.2**	84.1 / 51.6	86.4 / **44.9**	85.2 / 52.1	84.1 / 58.5	83.0 / 50.7
Dehydra$_{\text{Improved}}$	93.1 / **45.8**	93.1 / **5.7**	92.7 / **44.9**	91.4 / **41.5**	93.3 / **47.6**	86.0 / **49.3**	88.1 / **44.3**	88.2 / **46.3**	90.1 / **46.2**	88.5 / **47.2**

and 0.003 when data-free. Noteworthily, the settings above are empirically chosen and lead to strong attack effectiveness uniformly over almost all the covered watermarking schemes, which supports the watermark-agnostic nature of our Dehydra.

• **Evaluation Metrics.** We primarily focus on the utility and the watermark retention of the surrogate models obtained via removal attacks. Specifically, we use *clean accuracy*, i.e., the classification accuracy on the test set, to measure the utility, and *watermark accuracy*, i.e., the ratio of watermark samples correctly classified as target labels, to measure the watermark retention. Further, we follow Lukas et al. [42] to determine the decision threshold θ for each watermark on 20 clean models, and then calculate a metric $S(\cdot;\theta)$ called the *rescaled watermark accuracy*. The former θ is used to distinguish the watermark accuracy of the watermarked model from those independently trained models with high confidence, while the latter $S(\cdot;\theta)$ enables a fair robustness comparison over different watermarking schemes. Specially, the rescaling function is defined as $S(x;\theta) = \max(0, \frac{1-\theta'}{1-\theta}x + \frac{\theta'-\theta}{1-\theta})$, linearly rescaling the watermark accuracy x based on the decision threshold θ. By manually picking a rescaled decision threshold $\theta' = 0.5$, a watermark is said to be removed if the rescaled watermark accuracy is lower than 50%. Table 12 in Appendix B.3 summarizes our estimated decision threshold for the ten black-box watermarking schemes.

We define a removal attack as successful (i.e., cracks the target watermark scheme) if the rescaled watermark accuracy is below 50% and the surrogate model maintains at least 90% of the original utility. Note that this criterion is slightly different from [42] where a maximum 5% utility loss is used. This is mainly because we consider the more realistic data-limited settings (e.g., 2% of the source training set in most experiments), while [42] assume the attacker has access to over 30% of the training set. Nevertheless, we also evaluate the attack performance when more data is available in Section 7.

6.2 Attack Performance

The comparison with baseline removal attacks is organized according to the data settings where the baseline attacks are applicable. Due to the page limit, we mainly present and analyze the attack results on CIFAR-10, while similar results are observed on MNIST and CIFAR-100. Appendix C.2 and C.3 present the omitted results.

(1) In-distribution Setting. Table 2 presents the attack results by performing different removal attacks against the ten investigated watermarks under the in-distribution setting. Both the basic Dehydra and improved Dehydra significantly lower the rescaled watermark accuracy of surrogate models, among which the improved Dehydra achieves comprehensive watermark removal against the ten investigated black-box watermarks, with at least 90% utility preserved (six in ten even with a clean accuracy degradation in 2%).

The baseline removal attacks are only effective against a subset of watermark schemes. For instance, *Laundering* is more effective against watermarks similar to backdoors, such as *Content*, *Piracy* and *EWE*, but less effective against other watermarks, because of its dependence on the trigger reverse-engineering method to detect and remove the underlying watermarks. *Fine-pruning* and *Finetuning* are less effective against *EWE*, *Adi* and *AFS*, possibly due to the more entangled learned representations of the watermark data. *Distraction* can also only compromise about half of these watermarks' robustness, while being less powerful against watermarks of other forms. In contrast, our basic and improved Dehydra are both effective regardless of the target watermark schemes, due to the capture of the black-box watermarks' commonality. Note that our improved Dehydra can effectively remove the *Piracy* watermark, despite the fact that the number of its watermark data (1% of the training data, i.e., 500, following the original paper) is much larger than the number of unlearned samples in our improved Dehydra ($\gamma \times M = 125$). This indicates that the removal effect of our approach is achieved not just through a sample-level unlearning, but also through a deeper neuron-level fixing.

Additionally, these baseline attacks might impact the surrogate model's utility seriously. For instance, the finetuning-based attacks, such as *Finetuning*, *Regularization* and *Distraction*, incur a large clean accuracy degradation against *Noise* and *Piracy* watermarks, possibly due to their overly offensive finetuning strategies. However, our Dehydra attacks could still preserve the surrogate model's performance besides removing the watermark, because of their more specific training (unlearning) objectives.

(2) Transfer Setting. As we can see from Table 3, the clean accuracy of the surrogate models are generally lower than the in-distribution setting, since the proxy labels of these transfer data may contain noise. *Regularization* is generally ineffective, and *Distraction* is only effective against a subset of fixed-class watermarks.

On the contrary, both the basic and improved Dehydra significantly lower the rescaled watermark accuracy while preserving the clean accuracy at an acceptable level. The improved Dehydra

Table 3: Comparison of removal attacks against black-box watermarks under the transfer setting on CIFAR-10.

Attacks	*Content*	*Noise*	*Unrelated*	*Piracy*	*EWE*	*Adi*	*AFS*	*EW*	*Blind*	*Mark*
None	93.8 / 100.0	94.1 / 100.0	93.7 / 100.0	93.3 / 100.0	94.0 / 100.0	93.9 / 100.0	93.0 / 100.0	92.9 / 100.0	91.2 / 100.0	94.0 / 100.0
Regularization	49.3 / 99.4	60.3 / 73.6	53.1 / 100.0	60.5 / 100.0	53.8 / **47.6**	41.0 / 69.7	43.4 / 71.5	55.5 / 76.9	44.6 / 69.6	53.4 / 74.5
Distraction	89.0 / 50.9	89.6 / 100.0	88.7 / **49.8**	84.2 / **49.0**	88.8 / 63.4	87.7 / 60.3	92.0 / 100.0	92.3 / 100.0	88.8 / 91.2	89.1 / 66.9
Dehydra$_{\text{Basic}}$	85.5 / **49.8**	85.3 / **6.7**	80.0 / **46.0**	82.4 / **45.2**	88.2 / 50.8	80.8 / 54.5	84.6 / **48.1**	82.1 / 58.5	80.2 / 62.0	84.0 / 55.9
Dehydra$_{\text{Improved}}$	92.5 / **47.0**	92.3 / **5.7**	92.7 / **48.2**	90.1 / **47.1**	92.6 / **49.7**	80.4 / 59.8	86.2 / **47.4**	83.5 / **49.8**	88.8 / **46.8**	87.0 / **49.5**

Table 4: Comparison of removal attacks against black-box watermarks under the data-free setting on CIFAR-10.

Attacks	*Content*	*Noise*	*Unrelated*	*Piracy*	*EWE*
None	93.8 / 100.0	94.1 / 100.0	93.7 / 100.0	93.3 / 100.0	94.0 / 100.0
Pruning	81.5 / 78.2	90.7 / **42.5**	78.7 / **45.4**	41.9 / **37.8**	92.2 / 100.0
Dehydra (Improved)	89.8 / **45.8**	88.6 / **5.7**	88.8 / **46.0**	90.0 / **39.6**	90.7 / **48.2**

still successfully cracks nine of the ten watermarks under this setting. Note that our Dehydra cannot completely remove the *Adi* watermark (despite the rescaled watermark accuracy being lower than 60%) under the transfer setting. This is reasonable because *Adi* uses abstract out-of-distribution images as watermark data, whose salient neurons are more likely to be activated by the transfer dataset, compared to other common watermarks.

(3) Data-free Setting. Almost no baseline attacks except for the *Pruning* attack are applicable to the data-free scenario. Therefore, as also explained in Section 5.2, we mainly compare the improved Dehydra, when attacking five fixed-class watermarks here, with *Pruning*. The attack results are shown in Table 4.

As is shown, our improved Dehydra achieves surprisingly good performance in the data-free setting, significantly surpassing the baseline *Pruning*. All the five fixed-class watermarks are completely removed from the protected model, with a clean accuracy degradation of no more than 5.5%. This validates the correctness of our splitting algorithm on the recovered samples, since the surrogate modes are finetuned solely on the recovered samples $\{B_c\}_* = \{B_{s_0}\}$, with split proxy normal data $\{B_{s_0,nor}\}$ to maintain the utility, and proxy watermark data $\{B_{s_0,wmk}\}$ for removal, as in Equation 7.

6.3 Validating Detection of Target Classes

In this part, we mainly study the risks of target class detection errors. We temporarily ignore the case when fixed-class watermarks are correctly identified but given the incorrect target label, since this is extremely rare, as validated by later quantitative experiments. Here we mainly focus on two types of errors, false positives (non-fixed-class watermarks identified as fixed-class ones) and false negatives (fixed-class watermarks identified as non-fixed-class ones).

Intuitively, we have a lower tolerance for false positives compared to false negatives. As shown in Table 5, for fixed-class watermarks (*Content* and *Piracy*) wrongly identified as non-fixed-class ones, the clean accuracy will degrade a little (e.g., within 3% on CIFAR-10) and the watermark is still effectively removed. However, for non-fixed-class watermarks (*Adi* and *AFS*) falsely identified as fixed-class ones, the removal effectiveness will be significantly hampered (e.g, the rescaled watermark accuracy remains over 90%). Therefore, we should set the detection threshold conservatively to prevent false positive cases.

Table 5: Attack results of improved Dehydra on CIFAR-10 and CIFAR-100, with the target class detection results deliberately *resp.* set to fixed-class and non-fixed-class ones.

Dataset	Detection	*Content*	*Piracy*	*Adi*	*AFS*
CIFAR-10	fixed	93.1 / **45.8**	91.4 / **41.5**	92.0 / 96.5	88.8 / 93.7
	non-fixed	90.7 / **44.2**	88.9 / **40.3**	86.0 / **49.3**	88.1 / **44.3**
CIFAR-100	fixed	75.1 / **48.7**	71.0 / **41.8**	64.0 / 71.5	67.1 / 100.0
	non-fixed	67.7 / **48.7**	65.8 / **40.0**	64.8 / 60.3	65.4 / 54.8

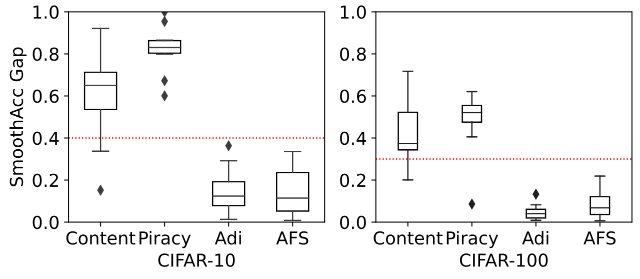

Figure 5: SmoothAcc gap distribution under ten random tests. The box plot shows min/max and quartiles, as well as the estimated outliers.

Next, we further investigate the detection accuracy of our target class detection algorithm. For each watermark scheme on each dataset, we independently train ten watermarked models, each using the randomly re-generated watermark data with different target classes. For each target watermarked model, we perform watermark recovering and then target class detection. We calculate the SmoothAcc gap $SmoothAcc(B_{y_w}) - \max_{i \neq y_w} SmoothAcc(B_i)$ for a fixed-class watermark with target class y_w, and $SmoothAcc(B_{s_1}) - SmoothAcc(B_{s_2})$ for a non-fixed-class watermark. As shown in Figure 5, the SmoothAcc gaps for fixed-class watermarks are always positive in the ten tests, which means we can always determine a correct target class for those successfully identified fixed-class watermarks. Besides, the SmoothAcc gaps of fixed-class and non-fixed-class watermarks are large enough to find a threshold to determine them, which validates the correctness of our target class detection algorithm.

6.4 Ablation Studies

6.4.1 Effectiveness of the Two Improvements. We further validate the effectiveness of the two proposed improvements. As shown in Table 6, target class detection is especially useful to fixed-class watermarks. It is also useful to non-fixed-class watermarks, due to the easier unlearning target of proxy hard labels. The splitting algorithm is generally beneficial to both fixed-class and non-fixed-class watermarks to preserve normal performance, due to the enhanced

Table 6: Ablation study on the two improvements, target class detection and recovered samples splitting, under the in-distribution setting on CIFAR-10.

Attacks	*Unrelated*	*EWE*	*EW*	*Mark*
Dehydra$_{\text{Basic}}$	88.1 / **44.9**	89.0 / **48.2**	85.2 / 52.1	83.0 / 50.7
Dehydra + Detection	91.9 / **44.9**	92.4 / **47.6**	86.8 / **46.9**	87.8 / **48.4**
Dehydra + Splitting	88.3 / **44.9**	89.6 / **47.6**	87.2 / 51.5	85.7 / **48.9**
Dehydra$_{\text{Improved}}$	92.7 / **44.9**	93.3 / **47.6**	88.2 / **46.3**	88.5 / **47.2**

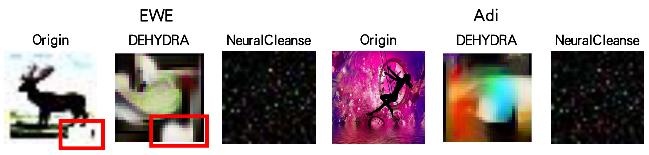

Figure 6: Visualization of the recovered watermark images/trigger patterns by Dehydra/*NeuralCleanse*.

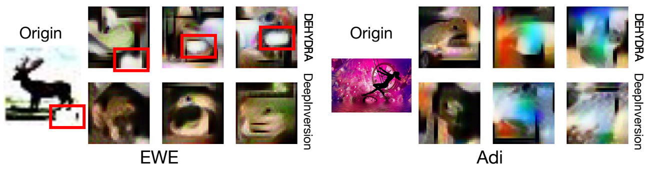

Figure 7: Visualization of the recovered samples by Dehydra and *DeepInversion*.

Table 7: Attack results using different recovering methods on CIFAR-10.

Inversion Method	*Content*	*EWE*	*Adi*
Dehydra	92.6 / **44.7**	92.5 / **48.2**	85.1 / **48.7**
NeuralCleanse	92.5 / **47.5**	92.2 / 98.4	84.4 / 93.0
DeepInversion	91.3 / 68.7	91.2 / 99.0	84.3 / **49.3**

removal specificity. Armed with the two designs, the improved Dehydra achieves the highest clean accuracy, with a 4% increase in average compared with the basic version.

6.4.2 Comparison with Other Recovering Methods. We further prove the effectiveness of the watermark recovering algorithm of Dehydra, by comparing with representative algorithms from **(1) Trigger Reverse-Engineering** (e.g., an adaptive version of *NeuralCleanse* [56]), which is originally designed to detect and erase backdoors in DNNs, and **(2) General Model Inversion** (e.g., *DeepInversion* [61]), which is designed to synthesize realistic training samples of the target models for privacy disclosure or knowledge distillation. Our experiments below focus on the in-distribution data setting on CIFAR-10, and perform watermark removal attacks against three black-box watermarks, *Content*, *EWE* and *Adi*. More implementation details are deferred to Appendix B.5.

Visual Effects. Figure 6 visualizes the recovered watermark data by Dehydra and the trigger patterns by *NeuralCleanse*, while Figure 7 compares Dehydra with *DeepInversion*. As is shown, the recovered triggers by *NeuralCleanse* are visually meaningless and rather similar to each other. This implies that the assumed shortcut in the model's decision space is non-existent in these watermarked models, and the recovered triggers are only the universal adversarial perturbations of the CIFAR-10 dataset itself. Also, *DeepInversion* tends to produce more realistic images belonging to the target classes (e.g., "frogs" for *EWE*), but less informative of the watermark data. In comparison, most of the recovered watermark data holds the essential features of the real watermark patterns, such as the white patches for *EWE* and colorful fringing for *Adi*. These visual differences demonstrate the effectiveness of our aggressive inversion scheme.

Attack Results. More quantitatively, we investigate the removal effectiveness using the three recovering methods, with attack results shown in Table 7. As we can see, *NeuralCleanse* is ineffective against *EWE* and *Adi*, due to the inexistence of backdoor shortcuts. For *DeepInversion*, unlearning the inverted samples from class 6 fails to remove *EWE*, because the BN statistics of the watermark data might diffuse into other classes during its random-class inversion scheme. The recovered samples are therefore more representative of the normal data, unlearning which incurs a larger utility loss. Only Dehydra is effective for removing all three watermarks from the protected models, conforming to the previous visual observations.

Table 8: Effectiveness of Dehydra on watermarked models after differential privacy training.

Eps	*Content*		*Adi*	
	Before	After	Before	After
$\epsilon = 1$	57.3 / 100.0	51.5 / 100.0	53.0 / 100.0	45.0 / 68.5
$\epsilon = 2$	62.0 / 100.0	55.9 / 100.0	60.0 / 99.4	53.9 / 62.1
$\epsilon = 6$	79.8 / 100.0	63.4 / 83.3	77.8 / 99.4	62.8 / 57.4

Table 9: The effectiveness of deceiving the detection algorithm against our improved Dehydra.

N_{trap}	*Adi*			*EW*		
	Before	After	Gap	Before	After	Gap
100	87.6 / 100.0	84.4 / 60.3	0.1233	90.2 / 100.0	87.4 / 51.5	0.0766
150	90.3 / 100.0	85.1 / 60.9	0.1291	92.7 / 100.0	89.9 / 92.5	0.7825
200	89.4 / 100.0	89.5 / 95.9	0.5406	90.3 / 100.0	90.0 / 98.3	0.6226

Table 10: The effectiveness of bypassing the splitting algorithm against our improved Dehydra.

WM size	*Content*		*EWE*	
	Before	After	Before	After
1000	91.9 / 100.0	89.0 / 52.0	90.3 / 99.5	89.2 / 62.3
2000	91.8 / 100.0	90.3 / 53.1	90.0 / 100.0	89.2 / 74.9
5000	88.5 / 100.0	87.9 / 58.1	89.4 / 100.0	88.6 / 79.6

6.5 Potential Defenses

The experiments above validate that Dehydra poses real threats to existing black-box watermarks. In this section, we further consider potential defenses against Dehydra. Specifically, we design defenses to resist or bypass the following three key components of Dehydra: the inversion-based framework, target class detection, and recovered samples splitting.

Differential Privacy. Differential privacy (DP) is a general approach for privacy protection [1]. Here, we investigate the effectiveness of differential privacy to reduce the watermark information

Table 11: Comparison to extraction attacks under different data ratios w.r.t. the training data on CIFAR-10.

Data Ratio	*Noise*			*EWE*			*AFS*		
	DEHYDRA	*Extraction*	*Combined*	DEHYDRA	*Extraction*	*Combined*	DEHYDRA	*Extraction*	*Combined*
2%	93.1 / **5.7**	46.5 / **24.6**	45.5 / **14.2**	93.3 / **47.6**	48.8 / 56.5	46.6 / 50.3	88.1 / **44.3**	48.0 / **44.9**	45.8 / **45.5**
5%	93.3 / **10.4**	62.1 / 88.7	56.6 / **5.7**	93.1 / **48.7**	58.7 / 55.0	56.0 / **49.2**	90.4 / **46.2**	63.5 / **46.2**	53.5 / **48.1**
10%	93.4 / **8.5**	72.4 / **18.0**	65.6 / **5.7**	93.3 / **49.7**	70.1 / **48.2**	66.3 / **49.2**	91.2 / **46.8**	71.8 / **46.8**	67.4 / **48.7**
33.33%	94.0 / **8.5**	86.3 / 81.1	78.5 / **33.1**	93.2 / **49.2**	81.8 / 60.2	78.9 / **47.6**	92.8 / 50.6	86.7 / **45.5**	78.5 / **50.0**

inverted by DEHYDRA. Roughly speaking, we use DP-SGD with the privacy budgets (ϵ, δ) during the watermark embedding process: δ is set to be a fixed value of 1×10^{-5} and the ϵ is varied to evaluate the attack effect under different privacy budgets, a smaller ϵ means adding larger noise to the gradients. As we can see in Table 8, with the decrease of ϵ, DEHYDRA will achieve a worse watermark removal. This is because the noise added to the gradients during the training phase makes the learned features of watermark information more scattered, thus compromising the effectiveness of the attack. However, this is brought with a significant accuracy drop of 40% on average.

Deceiving the Detection Algorithm. Here we try to cause the severer false positive errors (making non-fixed-class watermarks detected as fixed-class ones) in Section 6.3. Concretely, model owners can deliberately change the watermark smoothness by adding N_{trap} watermark samples with the specified identical labels (in addition to the original 100 watermark samples with non-fixed target classes) during training. In Table 9, when $N_{trap} = 200$, the SmoothAcc Gap will be larger than the detection threshold, and the watermark will be detected as fixed-class ones. However, forcing models to memorize more samples will cause an accuracy drop about 3% on average.

Bypassing the Splitting Algorithm. The model owner may attempt to bypass the recovered samples splitting algorithm, by deliberately adding more watermark samples into the target classes and disrupting the dominance of normal data. This may decrease the attack effectiveness of the improved DEHYDRA in data-limited settings, as we can see in Table 10, the larger trigger set size, the worse the removal effect will be. However, the attacker may skip the splitting procedure and restore the attack effectiveness with in-distribution/transfer data.

7 DISCUSSION

Black-box DNN Watermarks and Backdoors. In previous literature, black-box watermarks are sometimes also called backdoor-based [3, 39, 47] or trigger set-based watermarks [34]. However, black-box watermarks have substantial differences from DNN backdoors in terms of the watermark data patterns and target class settings (§2.1), especially for the non-fixed-class watermarks.

Therefore, attacking black-box watermarks has its unique challenges, especially for an attacker who usually has no knowledge of the underlying watermarking schemes. This also explains why existing removal attacks extended from backdoor defenses, such as *Laundering* (§6.2) and *NeuralCleanse* (§6.4.2), could not effectively remove most of the watermarks.

Attack Performance when More Data is Available. Our work primarily focuses on the data-limited settings, where only 2% of the training data is used in most of the experiments (§6.2). This limited data availability may have a slight impact on the utility of the surrogate model for several non-fixed-class watermarks. In this part, we evaluate the performance of the improved DEHYDRA when more data is available. As shown in Table 11, for the non-fixed-class watermark *AFS*, the utility loss is mitigated when more data is available. The accuracy drop is controlled within 1% for the three investigated watermarks, when 33% of the training data is available.

Comparison to Model Extraction Attacks. Model extraction attacks are effective for watermark removal but typically require a large query dataset. Here we compare the improved DEHYDRA with the classical *Extraction* attack [27] and the recently proposed *Combined* attack (which combines transfer learning with label smoothing) that has been reported as effective against all watermarks using one-third of the training data [42]. As shown in Table 11, the surrogate models obtained through extraction attacks demonstrate significantly dependent performance on the available dataset size, resulting in relatively low accuracy when the data ratio is 2%, 5%, and 10%. In contrast, our DEHYDRA maintains high accuracy while effectively removing the watermark across all data ratios.

Comparison to Data Extraction Attacks. Data extraction attacks aim at reconstructing training data from a well-trained DNN model. Since DNN watermarks are typically designed to trace ownership after model leakage (§2.1), it is essential to thoroughly assess the privacy risk of the watermark data when the model parameters are completely exposed. In fact, previous *Content* [63] and *EWE* [27] watermarks did investigate the watermark privacy risk, while using naive extraction methods, leading to the conclusion that their watermarks are "secure". On the contrary, DEHYDRA specially targets the overfitted watermark data, and hence adopts an aggressive class-wise inversion method to better exploit model internals. Ablation studies in Section 6.4.2 also demonstrate that our DEHYDRA captures more meaningful visual features of the watermark samples (Figure 6, 7) and achieves significantly better removal results (Table 7) compared to traditional data extraction methods.

Attack Efficiency. For a target model with a large number of class labels, the main computation cost of DEHYDRA comes from the class-wise watermark recovering process, which is proportional to the number of classes. Similar to the solutions in NeuralCleanse [56], we find parallel computation on multiple GPUs and early-stopping mechanisms also useful here. Moreover, we propose a multi-class inversion mechanism while optimizing a mixed data batch, where the BN term is calculated for samples of each class separately.

Limitations and Future Works. Our methodological design and evaluations primarily focus on the image classification task, which is consistent with the mainstream DNN watermarking research [33, 34, 42]. We note that there do exist some recent works extending black-box DNN watermarks to other domains, including self-supervised learning for encoders, graph-based tasks, and text-based tasks. For these new applications, we posit that the watermark information is still deeply embedded within the internals of the watermarked models, suggesting that Dehydra continues to be a promising and effective attack. The primary challenge to adapt the Dehydra attack lies in developing a suitable recovering method (for instance, prompting-based methods [8] could be utilized to recover the watermark information stored in a large language model). Nonetheless, we leave it as future work to explore the further effectiveness of Dehydra on other tasks.

8 CONCLUSION

In this paper, we identify a shared trait of existing black-box DNN watermarks: they exploit the over-parameterization of DNNs to especially memorize the watermark data correlated to the target classes. Based on this, we propose a novel watermark-agnostic removal framework, namely Dehydra, which exploits the model internals to recover and unlearn the underlying watermarks. With in-depth analysis, theoretical justifications and pilot studies, we further improve Dehydra with target class detection and recovered sample splitting algorithms, which help preserve the model utility and significantly reduce the dataset dependence. Extensive evaluations on various settings against ten mainstream black-box watermarks demonstrate that Dehydra is generally effective and has minimal utility impact, with much more relaxed or even no assumptions on the dataset availability.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work was supported in part by the National Key Research and Development Program (2021YFB3101200), National Natural Science Foundation of China (62472096, 62172104, 62172105, 62102093, 62102091, 62302101, 62202106). Min Yang is a faculty of Shanghai Institute of Intelligent Electronics & Systems and Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China. Mi Zhang and Min Yang are the corresponding authors.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 1615–1631.

[3] William Aiken, Hyoungshick Kim, and Simon S. Woo. 2020. Neural Network Laundering: Removing Black-Box Backdoor Watermarks from Deep Neural Networks. *Comput. Secur.* 106 (2020), 102277.

[4] Marco Allodi, Alberto Broggi, Domenico Giaquinto, Marco Patander, and Antonio Prioletti. 2016. Machine learning in tracking associations with stereo vision and lidar observations for an autonomous vehicle. In *2016 IEEE intelligent vehicles symposium (IV)*. IEEE, 648–653.

[5] Mrinal R Bachute and Javed M Subhedar. 2021. Autonomous driving architectures: insights of machine learning and deep learning algorithms. *Machine Learning with Applications* 6 (2021), 100164.

[6] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. 2021. Robustness may be at odds with fairness: An empirical study on class-wise accuracy. In *NeurIPS 2020 Workshop on Pre-registration in Machine Learning*. PMLR, 325–342.

[7] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX security symposium (USENIX security 19)*. 267–284.

[8] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*. 2633–2650.

[9] Xuxi Chen, Tianlong Chen, Zhenyu Zhang, and Zhangyang Wang. 2021. You are caught stealing my winning lottery ticket! Making a lottery ticket claim its ownership. *Advances in Neural Information Processing Systems* 34 (2021), 1780–1791.

[10] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. 2021. Refit: a unified watermark removal framework for deep learning systems with limited data. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 321–335.

[11] Xinyun Chen, Wenxiao Wang, Yiming Ding, Chris Bender, Ruoxi Jia, Bo Li, and Dawn Song. 2019. Leveraging unlabeled data for watermark removal of deep neural networks. In *ICML workshop on Security and Privacy of Machine Learning*. 1–6.

[12] Zhenzhu Chen, Shang Wang, Anmin Fu, Yansong Gao, Shui Yu, and Robert H. Deng. 2022. LinkBreaker: Breaking the Backdoor-Trigger Link in DNNs via Neurons Consistency Check. *IEEE Transactions on Information Forensics and Security* 17 (2022), 2000–2014.

[13] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2921–2926.

[14] The Turing Way Community. 2021. *Licensing Machine Learning models*. The Turing Way. https://book.the-turing-way.org/reproducible-research/licensing/licensing-ml.html Accessed on July 21, 2024.

[15] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2019. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 485–497.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[17] Hugging Face. 2024. *Hugging Face Models*. Retrieved April 10, 2024 from https://huggingface.co/models

[18] Zhijin Ge, Fanhua Shang, Hongying Liu, Yuanyuan Liu, and Xiaosen Wang. 2023. Boosting Adversarial Transferability by Achieving Flat Local Maxima. *arXiv preprint arXiv:2306.05225* (2023).

[19] Tristan Greene. 2018. *IBM came up with a watermark for neural networks*. Retrieved April 10, 2024 from https://thenextweb.com/news/ibm-came-up-with-a-watermark-for-neural-networks

[20] Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[21] Shangwei Guo, Tianwei Zhang, Han Qiu, Yi Zeng, Tao Xiang, and Yang Liu. 2020. Fine-tuning Is Not Enough: A Simple yet Effective Watermark Removal Attack for DNN Models. In *International Joint Conference on Artificial Intelligence*.

[22] Abdalraouf Hassan and Ausif Mahmood. 2018. Convolutional recurrent deep learning model for sentence classification. *Ieee Access* 6 (2018), 13949–13957.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[25] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems* 32 (2019).

[26] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. pmlr, 448–456.

[27] Hengrui Jia, Christopher A Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. 2021. Entangled Watermarks as a Defense against Model Extraction.. In *USENIX Security Symposium*. 1937–1954.

[28] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[29] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[30] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications* 32 (2020), 9233–9244.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[32] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. 1998. The MNIST Database of Handwritten Digits. *http://yann.lecun.com/exdb/mnist/* (1998).

[33] Isabell Lederer, Rudolf Mayer, and Andreas Rauber. 2023. Identifying Appropriate Intellectual Property Protection Mechanisms for Machine Learning Models: A Systematization of Watermarking, Fingerprinting, Model Access, and Attacks. *arXiv preprint arXiv:2304.11285* (2023).

[34] Suyoung Lee, Wonho Song, Suman Jana, Meeyoung Cha, and Sooel Son. 2022. Evaluating the robustness of trigger set-based watermarks embedded in deep neural networks. *IEEE Transactions on Dependable and Secure Computing* (2022).

[35] Huiying Li, Emily Wenger, Shawn Shan, Ben Y Zhao, and Haitao Zheng. 2019. Piracy resistant watermarks for deep neural networks. *arXiv preprint arXiv:1910.01226* (2019).

[36] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems* 31 (2018).

[37] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. 2019. How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN. *Proceedings of the 35th Annual Computer Security Applications Conference* (2019). https://api.semanticscholar.org/CorpusID:207847538

[38] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings 21*. Springer, 273–294.

[39] Xuankai Liu, Fengting Li, Bihan Wen, and Qi Li. 2021. Removing backdoor-based watermarks in neural networks with limited data. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 10149–10156.

[40] Yifan Lu, Wenxuan Li, Mi Zhang, Xudong Pan, and Min Yang. 2024. Neural Dehydration: Effective Erasure of Black-box Watermarks from DNNs with Limited Data. *arXiv preprint arXiv:2309.03466* (2024).

[41] Nils Lukas. 2022. *Watermark-Robustness-Toolbox*. Retrieved April 10, 2024 from https://github.com/dnn-security/Watermark-Robustness-Toolbox

[42] Nils Lukas, Edward Jiang, Xinda Li, and Florian Kerschbaum. 2022. Sok: How robust is image classification deep neural network watermarking?. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 787–804.

[43] Xinsong Ma, Zekai Wang, and Weiwei Liu. 2022. On the tradeoff between robustness and fairness. *Advances in Neural Information Processing Systems* 35 (2022), 26230–26241.

[44] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2015. Inceptionism: Going deeper into neural networks. (2015).

[45] Ryota Namba and Jun Sakuma. 2019. Robust watermarking of neural network with exponential weighting. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. 228–240.

[46] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep Face Recognition. In *British Machine Vision Conference*. https://api.semanticscholar.org/CorpusID:4637184

[47] Masoumeh Shafieinejad, Jiaqi Wang, Nils Lukas, and Florian Kerschbaum. 2019. On the Robustness of Backdoor-based Watermarking in Deep Neural Networks. *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security* (2019).

[48] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[49] Shichang Sun, Haoqi Wang, Mingfu Xue, Yushu Zhang, Jian Wang, and Weiqiang Liu. 2021. Detect and Remove Watermark in Deep Neural Networks via Generative Adversarial Networks. In *Information Security - 24th International Conference, ISC 2021, Virtual Event, November 10-12, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13118)*, Joseph K. Liu, Sokratis K. Katsikas, Weizhi Meng, Willy Susilo, and Rolly Intan (Eds.). Springer, 341–357.

[50] Zhichuang Sun, Ruimin Sun, Long Lu, and Alan Mislove. 2021. Mind Your Weight(s): A Large-scale Study on Insufficient Machine Learning Model Protection in Mobile Apps. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, Michael D. Bailey and Rachel Greenstadt (Eds.). USENIX Association, 1955–1972. https://www.usenix.org/conference/usenixsecurity21/presentation/sun-zhichuang

[51] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[52] Qi Tian, Kun Kuang, Kelu Jiang, Fei Wu, and Yisen Wang. 2021. Analysis and applications of class-wise robustness in adversarial training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1561–1570.

[53] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*. 601–618.

[54] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*. 269–277.

[55] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[56] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 707–723.

[57] Tianhao Wang and Florian Kerschbaum. 2021. Riga: Covert and robust white-box watermarking of deep neural networks. In *Proceedings of the Web Conference 2021*. 993–1004.

[58] Han Xu, Xiaorui Liu, Yaxin Li, Anil Jain, and Jiliang Tang. 2021. To be robust or to be fair: Towards fairness in adversarial training. In *International conference on machine learning*. PMLR, 11492–11501.

[59] Yifan Yan, Xudong Pan, Mi Zhang, and Min Yang. 2023. Rethinking White-Box Watermarks on Deep Learning Models under Neural Structural Obfuscation. In *32th USENIX security symposium (USENIX Security 23)*.

[60] Ziqi Yang, Hung Dang, and Ee-Chien Chang. 2019. Effectiveness of Distillation Attack and Countermeasure on Neural Network Watermarking. *ArXiv* abs/1906.06046 (2019).

[61] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8715–8724.

[62] Youngsik Yoon, Jinhwan Nam, Hyojeong Yun, Dongwoo Kim, and Jungseul Ok. 2022. Few-Shot Unlearning by Model Inversion. *arXiv preprint arXiv:2205.15567* (2022).

[63] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 159–172.

[64] Jianpeng Zhang, Yutong Xie, Qi Wu, and Yong Xia. 2019. Medical image classification using synergic deep learning. *Medical image analysis* 54 (2019), 10–19.

[65] Qi Zhong, Leo Yu Zhang, Shengshan Hu, Longxiang Gao, Jun Zhang, and Yang Xiang. 2022. Attention Distraction: Watermark Removal Through Continual Learning with Selective Forgetting. *2022 IEEE International Conference on Multimedia and Expo (ICME)* (2022), 1–6.