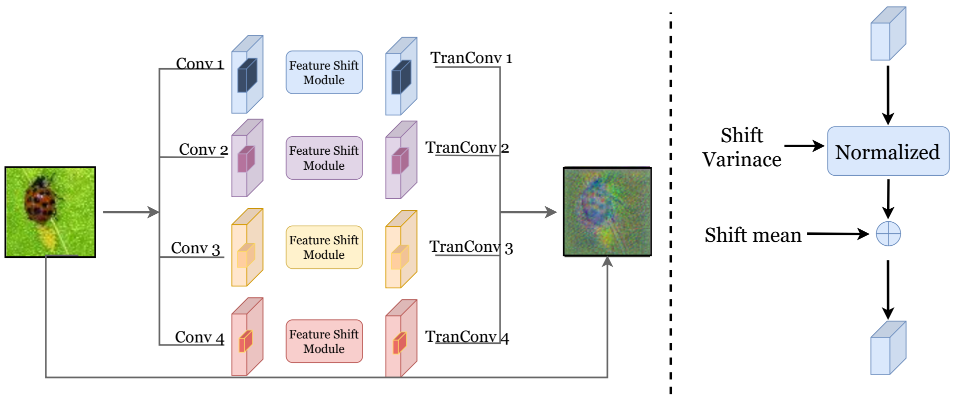
Appendix

Table of Contents



Overview of architectures for G_d Details of Feature Shift Module

Figure 1: The architecture of G_d.

A Technical Details for Generating Hardly-Generalized Domain

This process is mainly motivated by [33], which leveraged a transformation module with different convolution transformations to minimize the mutual information (I) between features from the source dataset ($i.e.$, $\boldsymbol{z}$) and data from the target domain ($i.e.$, $\hat{\boldsymbol{z}}$). Our work is also partially inspired by previous work on generating unlearnable samples [45], which crafted effective unlearnable samples by performing the bi-level optimization within each iteration.

A.1 The Implementation of Transformation Module

We follow previous work [33] to implement the transformation module for generating samples from a different domain (as shown in Fig. 1). Specifically, we design the transformation module as an ensemble of multiple ($i.e.$, 4) convolution operations. Each convolution operation contains a convolution layer `Conv`, a feature shift module, and a corresponding transposed convolution layer `TranConv`. The detailed parameters for each convolution layer $\texttt{Conv}_i$ are detailed in Tab. 1. Following each convolution layer $\texttt{Conv}_i$, we add a feature shift module to enhance the diversity of the generated samples. Specifically, each feature shift module contains two learnable parameters μ_i and σ_i as mean shift and variance shift, following:

$$\sigma_i \cdot \frac{\texttt{Conv}_i(\boldsymbol{x}) - \mu}{\sigma} + \mu_i, \tag{1}$$

where μ and σ represent the mean and covariance value for $\texttt{Conv}_i(\boldsymbol{x})$. Notably, μ and σ are not learnable parameters. Moreover, the parameters μ_i and σ_i has the same dimension as the output of $\texttt{Conv}_i(\boldsymbol{x})$. After that, we use a transposed convolution layer `TranConv` to turn the feature maps generated by the above operations into a real instance, which has the same dimension as $\boldsymbol{x}$.

Putting all above, we generate the hard-generalized domain samples $\hat{\boldsymbol{x}}$ following:

$$\hat{\boldsymbol{x}} = \frac{1}{\sum w_i} \sum_i w_i \cdot \text{tahn}(\texttt{TranConv}(\sigma_i \cdot \frac{\texttt{Conv}_i(\boldsymbol{x}) - \mu}{\sigma} + \mu_i)), \tag{2}$$

where tahn represents the tahn activation function. w_i is a scalar and weights the contribution of each activated instance produced by `TransposedConv` to $\hat{\boldsymbol{x}}$. w_i is randomly sampled from normal distribution $w_i \sim N(0, 1)$. Notably, for each input $\boldsymbol{x}$, we first up-sample it to 224×224 size and down-sample produced $\hat{\boldsymbol{x}}$ to the original size for $\boldsymbol{x}$.

A.2 The Optimization Process

During the optimization process of Eq. (2), we first initialized a surrogate model $f(\cdot; \boldsymbol{w})$ and a benign dataset $\mathcal{D}$. Then during each iteration for solving the bi-level optimization Eq. (2), we first minimize the $I(\boldsymbol{z}; \hat{\boldsymbol{z}})$ and $\mathcal{L}_c$ by optimizing the parameters of our proposed transformation module:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{z}, \hat{\boldsymbol{z}})} \left[I(\boldsymbol{z}(\boldsymbol{w}^*); \hat{\boldsymbol{z}}(\boldsymbol{\theta}, \boldsymbol{w}^*)) + \lambda_1 \mathcal{L}_c(\boldsymbol{z}(\boldsymbol{w}^*), \hat{\boldsymbol{z}}(\boldsymbol{\theta}, \boldsymbol{w}^*)) \right]. \tag{3}$$

Table 1: The configuration for each Convolution layer `Conv`.

Model	Kernel Size	Input Channel	Output Channel
$\mathtt{Conv}_1$	5x5	3	3
$\mathtt{Conv}_2$	9x9	3	3
$\mathtt{Conv}_3$	13x13	3	3
$\mathtt{Conv}_4$	17x17	3	3

After that, we maximize $I(\boldsymbol{z};\hat{\boldsymbol{z}})$ and minimize the training loss by optimizing the parameters $\boldsymbol{w}$:

$$\min_{\boldsymbol{w}} \left[\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\mathcal{L}(f(G_d(\boldsymbol{x};\boldsymbol{\theta});\boldsymbol{w}),y)+\mathcal{L}(f(\boldsymbol{x};\boldsymbol{w}),y)\right]-\lambda_2\mathbb{E}_{p(\boldsymbol{z},\hat{\boldsymbol{z}})}[I(\boldsymbol{z}(\boldsymbol{w});\hat{\boldsymbol{z}})\right]. \tag{4}$$

Since $I(\boldsymbol{z};\hat{\boldsymbol{z}})$ is intractable, we propose to optimize its upper bound instead:

$$I(\boldsymbol{z};\hat{\boldsymbol{z}})=\mathbb{E}_{p(\boldsymbol{z},\hat{\boldsymbol{z}})}\left[\log\frac{p(\hat{\boldsymbol{z}}|\boldsymbol{z})}{p(\hat{\boldsymbol{z}})}\right]\leq\mathbb{E}_{p(\boldsymbol{z},\hat{\boldsymbol{z}})}[\log p(\hat{\boldsymbol{z}}|\boldsymbol{z})]-\mathbb{E}_{p(\boldsymbol{z})p(\hat{\boldsymbol{z}})}[\log p(\hat{\boldsymbol{z}}|\boldsymbol{z})]. \tag{5}$$

Since the conditional distribution $p(\hat{\boldsymbol{z}}|\boldsymbol{z})$ is also intractable thus the upper bound of $I(\boldsymbol{z};\hat{\boldsymbol{z}})$ can't be optimized, we follow previous work to adopt a variational distribution $q(\hat{\boldsymbol{z}}|\boldsymbol{z})$ to approximate the upper bound of $I(\boldsymbol{z};\hat{\boldsymbol{z}})$:

$$I(\boldsymbol{z};\hat{\boldsymbol{z}})\leq\frac{1}{N}\sum_{i=1}^{N}[\log q(\hat{\boldsymbol{z}}_{\boldsymbol{i}}|\boldsymbol{z}_{\boldsymbol{i}})-\frac{1}{N}\sum_{j=1}^{N}\log q(\hat{\boldsymbol{z}}_{\boldsymbol{j}}|\boldsymbol{z}_{\boldsymbol{i}})], \tag{6}$$

where $q(\hat{\boldsymbol{z}}|\boldsymbol{z})$ is obtained by employing the backbone neural network to approximate.

We optimize the above bi-level optimization Eq. (2) with 100 iterations. We set the learning rate as 0.005 for optimizing the parameters of the proposed transformation module and 0.001 for parameters for the backbone model $f(\cdot)$ following [33]. The batch size is 64. For both the transformation module and the backbone model $f(\cdot)$, we use SGD [46] as the optimizer with Nesterov momentum and weight decay rate of 0.0005. We use ResNet-18 as the backbone model for extracting $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$ throughout the paper. We introduce λ_1 and λ_2 for balancing each optimization objective. Following the implementation of [33], we set λ_1 and λ_2 as 0.1 and 1.0 for balancing each optimization objective.

B The Proof for Theorem 1

Theorem 1 (Data Quantity Impact)**.** *Suppose in PAC Bayesian [35], for a target domain $\mathcal{T}$ and a source domain $\mathcal{S}$, any set of voters (candidate models) $\mathcal{H}$, any prior π over $\mathcal{H}$ before any training, any $\xi\in(0,1]$, any $c>0$, with a probability at least $1-\xi$ over the choices of $S\sim\mathcal{S}^{n_s}$ and $T\sim\mathcal{T}_{\mathcal{X}}^{n_t}$, for the posterior f over $\mathcal{H}$ after the joint training on S and T, we have*

$$\begin{aligned}\mathcal{R}_{\mathcal{T}}(f)\leq{}&\frac{c}{2(1-e^{-c})}\widehat{\mathcal{R}}_T(f)+\frac{c}{1-e^{-c}}\beta_\infty(\mathcal{T}\|\mathcal{S})\widehat{\mathcal{R}}_S(f)+\Omega\\&+\frac{1}{1-e^{-c}}\left(\frac{1}{n_t}+\frac{\beta_\infty(\mathcal{T}\|\mathcal{S})}{n_s}\right)\left(2\mathrm{KL}(f\|\pi)+\ln\frac{2}{\xi}\right),\end{aligned} \tag{7}$$

where $\widehat{\mathcal{R}}_T(f)$ and $\widehat{\mathcal{R}}_S(f)$ are the target and source empirical risks measured over target and source datasets T and S, respectively. Ω is a constant and $\mathrm{KL}(\cdot)$ is the Kullback–Leibler divergence. $\beta_\infty(\mathcal{T}\|\mathcal{S})$ is a measurement of discrepancy between $\mathcal{T}$ and $\mathcal{S}$ defined as

$$\beta_\infty(\mathcal{T}\|\mathcal{S})=\sup_{(\boldsymbol{x},y)\in\mathrm{SUPP}(\mathcal{S})}\left(\frac{\mathcal{P}_{(\boldsymbol{x},y)\in\mathcal{T}}}{\mathcal{P}_{(\boldsymbol{x},y)\in\mathcal{S}}}\right)\geq 1, \tag{8}$$

where $\mathrm{SUPP}(\mathcal{S})$ denotes the support of $\mathcal{S}$. When $\mathcal{S}$ and $\mathcal{T}$ are identical, $\beta_\infty(\mathcal{T}\|\mathcal{S})=1$.

Proof. Theorem 6 in Germain *et al.* 's work [47] demonstrates that *suppose in PAC Bayesian [35], for a target domain $\mathcal{T}$ and a source domain $\mathcal{S}$, any set of voters (candidate models) $\mathcal{H}$, any prior*

π over $\mathcal{H}$ before any training, any $\xi \in (0,1]$, any $c > 0$, with a probability at least $1-\xi$ over the choices of $S \sim \mathcal{S}^{n_s}$ and $T \sim \mathcal{T}_{\mathcal{X}}^{n_t}$, for the posterior f over $\mathcal{H}$ after the joint training on S and T:

$$\begin{aligned}\mathcal{R}_{\mathcal{T}}(f) \leq & \frac{c}{2(1-e^{-c})}\widehat{\mathrm{d}}_T(f) + \frac{c}{1-e^{-c}}\beta_\infty(\mathcal{T}\|\mathcal{S})\widehat{\mathrm{e}}_S(f) + \Omega \\ & + \frac{1}{1-e^{-c}}\left(\frac{1}{n_t} + \frac{\beta_\infty(\mathcal{T}\|\mathcal{S})}{n_s}\right)\left(2\mathrm{KL}(f\|\pi) + \ln\frac{2}{\xi}\right),\end{aligned} \tag{9}$$

where $\mathcal{R}_{\mathcal{T}}(f)$ denotes the expected Gibbs risk of voter f over the target domain. $\widehat{\mathrm{d}}_T(f)$ and $\widehat{\mathrm{e}}_S(f)$ are the empirical estimation of the target voters' disagreement and the source joint error, measured over target and source datasets T and S, respectively. Ω is a constant and $\mathrm{KL}(\cdot)$ is the Kullback–Leibler divergence. $\beta_\infty(\mathcal{T}\|\mathcal{S})$ is a measurement of discrepancy between $\mathcal{T}$ and $\mathcal{S}$ defined as

$$\beta_\infty(\mathcal{T}\|\mathcal{S}) = \sup_{(\boldsymbol{x},y)\in \mathrm{SUPP}(\mathcal{S})}\left(\frac{\mathcal{P}_{(\boldsymbol{x},y)\in\mathcal{T}}}{\mathcal{P}_{(\boldsymbol{x},y)\in\mathcal{S}}}\right), \tag{10}$$

where $\mathrm{SUPP}(\mathcal{S})$ denotes the support of $\mathcal{S}$.

In the following proof, in particular, the Gibbs risk $\mathcal{R}_{\mathcal{A}}(f)$, the voters' disagreement $\mathrm{d}_{\mathcal{A}}(f)$, and the joint error $\mathrm{e}_{\mathcal{A}}(f)$ of a certain domain $\mathcal{A}$ are defined as follows

$$\mathcal{R}_{\mathcal{A}}(f) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{A}}\mathop{\mathbb{E}}_{h\sim f}\mathbb{I}[h(\boldsymbol{x}) \neq y], \tag{11}$$

$$\mathrm{d}_{\mathcal{A}}(f) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{A}_{\mathcal{X}}}\mathop{\mathbb{E}}_{h\sim f}\mathop{\mathbb{E}}_{h'\sim f}\mathbb{I}\left[h(\boldsymbol{x}) \neq h'(\boldsymbol{x})\right], \tag{12}$$

$$\mathrm{e}_{\mathcal{A}}(f) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{A}}\mathop{\mathbb{E}}_{h\sim f}\mathop{\mathbb{E}}_{h'\sim f}\mathbb{I}\left[h(\boldsymbol{x}) \neq y\right]\mathbb{I}\left[h'(\boldsymbol{x}) \neq y\right], \tag{13}$$

where $\mathbb{I}[\mathrm{True}] = 1$ if the inner condition is true, and otherwise $\mathbb{I}[\mathrm{False}] = 0$, and $\mathcal{A}_{\mathcal{X}}$ is the marginal distribution of domain $\mathcal{A}$. h and h' are votes sampled from the posterior distribution f over $\mathcal{H}$. With these definitions, studies [48, 49] reveal a relationship among the Gibbs risk, the voters' disagreement, and the joint error as

$$\mathcal{R}_{\mathcal{A}}(f) = \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{A}}\mathop{\mathbb{E}}_{h\sim f}\mathop{\mathbb{E}}_{h'\sim f}\frac{\mathbb{I}\left[h(\boldsymbol{x}) \neq h'(\boldsymbol{x})\right] + 2\mathbb{I}\left[h(\boldsymbol{x}) \neq y \wedge h'(\boldsymbol{x}) \neq y\right]}{2} = \frac{1}{2}\mathrm{d}_{\mathcal{A}}(f) + \mathrm{e}_{\mathcal{A}}(f). \tag{14}$$

In this case, we can extend this relationship to the empirical estimations (suppose a dataset A is sampled from domain $\mathcal{A}$) as

$$\widehat{\mathcal{R}}_A(f) = \frac{1}{|A|}\sum_{(\boldsymbol{x},y)\sim A}\mathop{\mathbb{E}}_{h\sim f}\mathop{\mathbb{E}}_{h'\sim f}\frac{\mathbb{I}\left[h(\boldsymbol{x}) \neq h'(\boldsymbol{x})\right] + 2\mathbb{I}\left[h(\boldsymbol{x}) \neq y \wedge h'(\boldsymbol{x}) \neq y\right]}{2} = \frac{1}{2}\widehat{\mathrm{d}}_A(f) + \widehat{\mathrm{e}}_A(f). \tag{15}$$

Then we can use $\widehat{\mathcal{R}}_T(f)$ and $\widehat{\mathcal{R}}_S(f)$ to replace $\widehat{\mathrm{d}}_T(f)$ and $\widehat{\mathrm{e}}_S(f)$ in Eq. (9), respectively. In the end, we can follow Xu *et al*. [50] to regard these empirical risks as data quantity-irrelevant when analyzing the impact of data quantity.

Next, we focus on the proof of the numerical relationship $\beta_\infty(\mathcal{T}\|\mathcal{S}) \geq 1$. First of all, $\beta_\infty(\mathcal{T}\|\mathcal{S})$ comes from a more general definition that is parameterized by a real value $q > 0$, shown as

$$\beta_q(\mathcal{T}\|\mathcal{S}) = \left[\mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{S}}\left(\frac{\mathcal{P}_{(\boldsymbol{x},y)\in\mathcal{T}}}{\mathcal{P}_{(\boldsymbol{x},y)\in\mathcal{S}}}\right)^q\right]^{\frac{1}{q}}. \tag{16}$$

For any $q > 0$, $\beta_q(\mathcal{T}\|\mathcal{S})$ can be also written as a Ŕenyi Divergence-based form [47], $i.e.$,

$$\beta_q(\mathcal{T}\|\mathcal{S}) = 2^{\frac{q-1}{q}D_q(\mathcal{T}\|\mathcal{S})}, \tag{17}$$

where $D_q(\mathcal{T}\|\mathcal{S})$ is the Ŕenyi Divergence between $\mathcal{T}$ and $\mathcal{S}$ with the order q. For Ŕenyi Divergence with any order $q > 0$, there is a property of positivity [51], $i.e.$, $D_q(\mathcal{T}\|\mathcal{S}) \geq 0$. In this case, when $q \to \infty$, Eq. (17) becomes $\beta_q(\mathcal{T}\|\mathcal{S}) = 2^{D_q(\mathcal{T}\|\mathcal{S})} \geq 1$, and $\beta_q(\mathcal{T}\|\mathcal{S}) = 2^{D_q(\mathcal{T}\|\mathcal{S})} = 1$ when $\mathcal{T} = \mathcal{S}$, in other words, $D_q(\mathcal{T}\|\mathcal{S}) = 0$ when $\mathcal{T} = \mathcal{S}$ [51]. □

C Technical Details for Generating Protected Dataset

C.1 The Optimization Solution for Generating Protected Dataset

Recall Eq. (7) is :

$$\min_{\boldsymbol{\delta} \subset \mathcal{B}} \left[\mathbb{E}_{(\hat{\boldsymbol{x}},y)\sim\mathcal{T}}[\mathcal{L}\left(f(\hat{\boldsymbol{x}};\boldsymbol{w}(\boldsymbol{\delta})),y\right)] - \lambda_3 \min\left\{ \mathbb{E}_{(\overline{\boldsymbol{x}},y)\sim\overline{\mathcal{T}}}[\mathcal{L}\left(f(\overline{\boldsymbol{x}};\boldsymbol{w}(\boldsymbol{\delta})),y\right)], \lambda_4 \right\} \right], \tag{18}$$

$$s.t.\ \boldsymbol{w}(\boldsymbol{\delta}) = \arg\min_{\boldsymbol{w}} \left[\frac{1}{|\mathcal{D}_s|} \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_s} \mathcal{L}\left(f(\boldsymbol{x}_i+\boldsymbol{\delta}_i;\boldsymbol{w}),y_i\right) + \frac{1}{|\mathcal{D}_b|} \sum_{(\boldsymbol{x}_j,y_j)\in\mathcal{D}_b} \mathcal{L}\left(f(\boldsymbol{x}_j;\boldsymbol{w}),y_j\right) \right],$$

where $\mathbb{E}_{(\overline{\boldsymbol{x}},y)\sim\overline{\mathcal{T}}}[\mathcal{L}\left(f(\overline{\boldsymbol{x}};\boldsymbol{w}(\boldsymbol{\delta})),y\right)]$ represents the expected risk for the watermarked model on other unseen domains ($i.e.$,$\overline{\mathcal{T}}$) and $\mathcal{B} = \{\boldsymbol{\delta} : ||\boldsymbol{\delta}||_\infty \le \epsilon\}$ where ϵ is a visibility-related hyper-parameter.

The aforementioned problem is a standard bi-level problem, we following previous work [52, 53] to leverage *gradient matching* to solving it. Specifically, we first make the following definition:

$$\mathcal{L}_t = \mathbb{E}_{(\hat{\boldsymbol{x}},y)\sim\mathcal{T}}[\mathcal{L}\left(f(\hat{\boldsymbol{x}};\boldsymbol{w}),y\right)] - \lambda_3 \min\left\{ \mathbb{E}_{(\overline{\boldsymbol{x}},y)\sim\overline{\mathcal{T}}}[\mathcal{L}\left(f(\overline{\boldsymbol{x}};\boldsymbol{w}),y\right)], \lambda_4 \right\}, \tag{19}$$

$$\mathcal{L}_i = \frac{1}{|\mathcal{D}_s|} \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_s} \mathcal{L}\left(f(\boldsymbol{x}_i+\boldsymbol{\delta}_i;\boldsymbol{w}),y_i\right). \tag{20}$$

According to the gradient-matching technique [52, 53], we have the Upper-level Sub-problem as:

$$\max_{\boldsymbol{\delta} \subset \mathcal{B}} \frac{\bigtriangledown_{\boldsymbol{w}}\mathcal{L}_t \cdot \bigtriangledown_{\boldsymbol{w}}\mathcal{L}_i}{||\bigtriangledown_{\boldsymbol{w}}\mathcal{L}_t|| \cdot ||\bigtriangledown_{\boldsymbol{w}}\mathcal{L}_i||}, \tag{21}$$

where we aim to maximize the gradient matching degree between $\bigtriangledown_{\boldsymbol{w}}\mathcal{L}_t$ and $\bigtriangledown_{\boldsymbol{w}}\mathcal{L}_i$ using $\texttt{cosine}(\cdot)$ similarity as the metric through optimizing δ. We solve the above Upper-level Sub-problem via projected gradient ascend (PGA). We here use calculate $\mathbb{E}_{(\hat{\boldsymbol{x}},y)\sim\mathcal{T}}[\mathcal{L}\left(f(\hat{\boldsymbol{x}};\boldsymbol{w}),y\right)]$ following:

$$\mathbb{E}_{(\hat{\boldsymbol{x}},y)\sim\mathcal{T}}[\mathcal{L}\left(f(\hat{\boldsymbol{x}};\boldsymbol{w}),y\right)] = \frac{1}{N} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \mathcal{L}(f(G_d(\boldsymbol{x});\boldsymbol{w}),y). \tag{22}$$

Regarding the Lower-level Sub-problem, we have:

$$\min_{\boldsymbol{w}} \left[\frac{1}{|\mathcal{D}_s|} \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_s} \mathcal{L}\left(f(\boldsymbol{x}_i+\boldsymbol{\delta}_i;\boldsymbol{w}),y_i\right) + \frac{1}{|\mathcal{D}_b|} \sum_{(\boldsymbol{x}_j,y_j)\in\mathcal{D}_b} \mathcal{L}\left(f(\boldsymbol{x}_j;\boldsymbol{w}),y_j\right) \right]. \tag{23}$$

After obtaining the poisoned dataset ($i.e.$, $\mathcal{D}_s \cup \mathcal{D}_b$), we can optimize the model ($i.e.$, ResNet-18) parameters $\boldsymbol{w}$ via solving the above Lower-level Upper-sub problem. The above Lower-level Upper-sub problem is solved via stochastic gradient descent.

We optimize the Upper-level and Lower-level Sub-problems alternatively for each optimization iteration. Specifically, we first train the model under benign dataset $\mathcal{D}$. Then for each iteration, we first optimize the Upper-level Sub-problem based on the trained model and obtain the perturbation δ. After that, we optimize the Lower-level Sub-problem based on the obtained poisoned dataset. During each iteration for optimizing the above bi-level optimization problems, we optimize the Upper-level Sub-problem with 50 iterations, and optimize the Lower-level Sub-problem with 100 iterations. We optimize the entire bi-level optimization with five epochs. The other details for optimization hyper-parameters as well as configuration are consistent with [53, 23].

In particular, to ensure the effectiveness of solving the aforementioned bi-level optimization problem, we have two additional strategies, as follows:

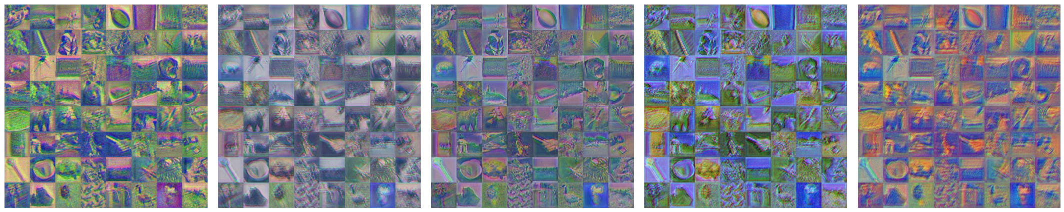


Figure 2: The example of samples generated from various domain.

- **Strategy 1:** Instead of randomly selecting samples from benign dataset $\mathcal{D}$, we here choose to select training samples with the largest gradient norms, following the previous work [52].
- **Strategy 2:** Instead of selecting samples from all classes, we follow the previous work [4] to select those from a specific class and the selected class is set as the target label. This strategy can enhance the effectiveness for solving the above bi-level optimization problem while preserving the verification performance for our approach.

C.2 The Process of Generating Samples from Other Domains

In this part, we describe how to generate samples from other domains ($i.e.$, $(\overline{\boldsymbol{x}}, y) \sim \overline{\mathcal{T}}$).

After obtaining the transformation module $G_d(\cdot)$, we can generate hard-generalized domain samples from a specific domain. We here propose to generate samples from other domains by setting different configurations of $\{w_i\}_1^4$. For example, we can generate samples from the other domain by sampling $\{w_i\}_1^4$ with another values following $w_i \sim N(0, 1)$.

We here show some demonstration of samples from other domains in Fig. 2.

We here generated samples from other domains, and estimate $\mathbb{E}_{(\overline{\boldsymbol{x}}, y)\sim\overline{\mathcal{T}}}[\mathcal{L}\left(f(\overline{\boldsymbol{x}}; \boldsymbol{w}), y\right)]$ following:

$$\mathbb{E}_{(\overline{\boldsymbol{x}}, y)\sim\overline{\mathcal{T}}}[\mathcal{L}\left(f(\overline{\boldsymbol{x}}; \boldsymbol{w}(\boldsymbol{\delta})), y\right)] = \frac{1}{N}\frac{1}{J}\sum_{j}\sum_{(\overline{\boldsymbol{x}}, \boldsymbol{y})\in\overline{\mathcal{T}}_j}\mathcal{L}(f(\overline{\boldsymbol{x}}; \boldsymbol{w}), y), \tag{24}$$

where $\overline{\mathcal{T}}_j$ represents the i-th unseen domain generated by the above approach.

C.3 The Selection of Hyper-parameters

After generating other unseen domains $\mathcal{T}$, we here describe the selection of hyper-parameters ($i.e.$, J and λ_3) for generating protected dataset.

We here propose a heuristic approach for selecting J and λ_3. Specifically, we first keep λ_3 fixed ($i.e.$,1) and adjust J. We conduct empirical study on CIFAR-10 tasks, the results are shown in Fig. 3.

We use ResNet-18 as the evaluated model. We generate several unseen domains using the above approach. We randomly select J of these domains for optimizing the Eq. (7), and select 3 unseen domains as the validation data. Notably, the validation domains are ensured visually different from the domains used for optimization.

From Fig. 3, we find that using $\geq$ three unseen domains is sufficient to constrain the generalization performance for validation unseen domains. Therefore, we set J as 3 for our approach.

After that, we keep J fixed, and adjust λ_3 gradually, the results are shown in Fig. 4. We find that when λ_3 becomes smaller, the constraint for performance on other unseen domains reduces. Accordingly, we set λ_3 as 0.3 for our approach since it can achieve a close generalization capacity compared to the benign DNN model ($i.e.$, 24.3%).

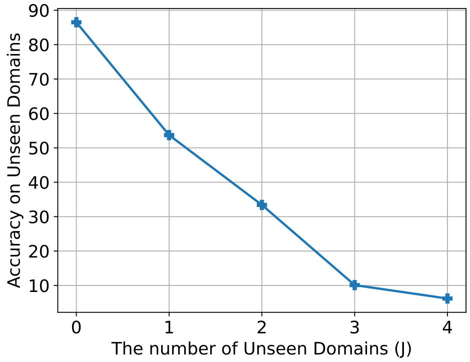


Figure 3: Effects of the number of Unseen Domains J.

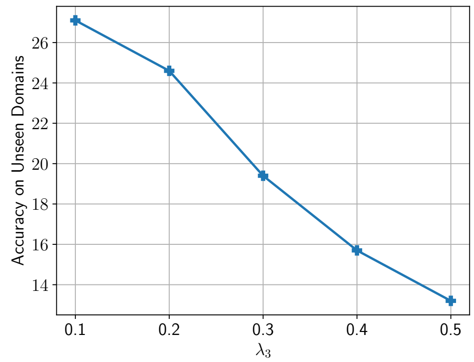


Figure 4: Effects of the λ_3.

D The Proof for Theorem 2

Theorem 2. *Let $f(\boldsymbol{x})$ is the posterior probability of $\boldsymbol{x}$ predicted by the suspicious model, variable $\boldsymbol{X}$ denotes the benign sample with label Y, and variable $\boldsymbol{X}'$ is the domain-watermarked version of $\boldsymbol{X}$. Assume that $P_b \triangleq f(\boldsymbol{X})_Y > \eta$. We claim that dataset owners can reject the null hypothesis H_0 at the significance level α, if the verification success rate (VSR) V of f satisfies that*

$$\sqrt{m-1} \cdot (V - \eta + \tau) - t_\alpha \cdot \sqrt{V - V^2} > 0, \tag{25}$$

where t_α is the α-quantile of t-distribution with $(m-1)$ degrees of freedom and m is the sample size.

Proof. Since $\boldsymbol{P}_b > \eta$, the original hypothesis H_1 can be converted to

$$H_1' : P_d > \eta - \tau. \tag{26}$$

Let E indicates the event of whether the suspect model f predicts a watermark sample as its ground-truth label y. As such, $E \sim B(1, p)$, where $p = \Pr(C(\boldsymbol{X}') = Y)$ indicates the verification success probability and B is the Binomial distribution [36].

Let $\hat{\boldsymbol{x}}_1, \cdots, \hat{\boldsymbol{x}}_m$ denotes m domain-watermarked samples used for dataset verification and $E_1, \cdots, E_m$ denote their prediction events, we know that the verification success rate V satisfies

$$V = \frac{1}{m} \sum_{i=1}^{m} E_i, \tag{27}$$

$$V \sim \frac{1}{m} B(m, p). \tag{28}$$

According to the central limit theorem [36], the verification success rate V follows Gaussian distribution $\mathcal{N}(p, \frac{p(1-p)}{m})$ when m is sufficiently large. Similarly, $(P_d - \eta + \tau)$ also satisfies Gaussian distribution. Accordingly, we can construct the t-statistic as follows:

$$T \triangleq \frac{\sqrt{m}(W - \eta + \tau)}{s} \sim t(m-1), \tag{29}$$

where s is the standard deviation of $(V - \eta + \tau)$ and V, *i.e.*,

$$s^2 = \frac{1}{m-1} \sum_{i=1}^{m} (E_i - V)^2 = \frac{1}{m-1} (m \cdot V - m \cdot V^2). \tag{30}$$

To reject the hypothesis H_0 at the significance level α, we need to ensure that

$$\frac{\sqrt{m}(V - \eta + \tau)}{s} > t_\alpha, \tag{31}$$

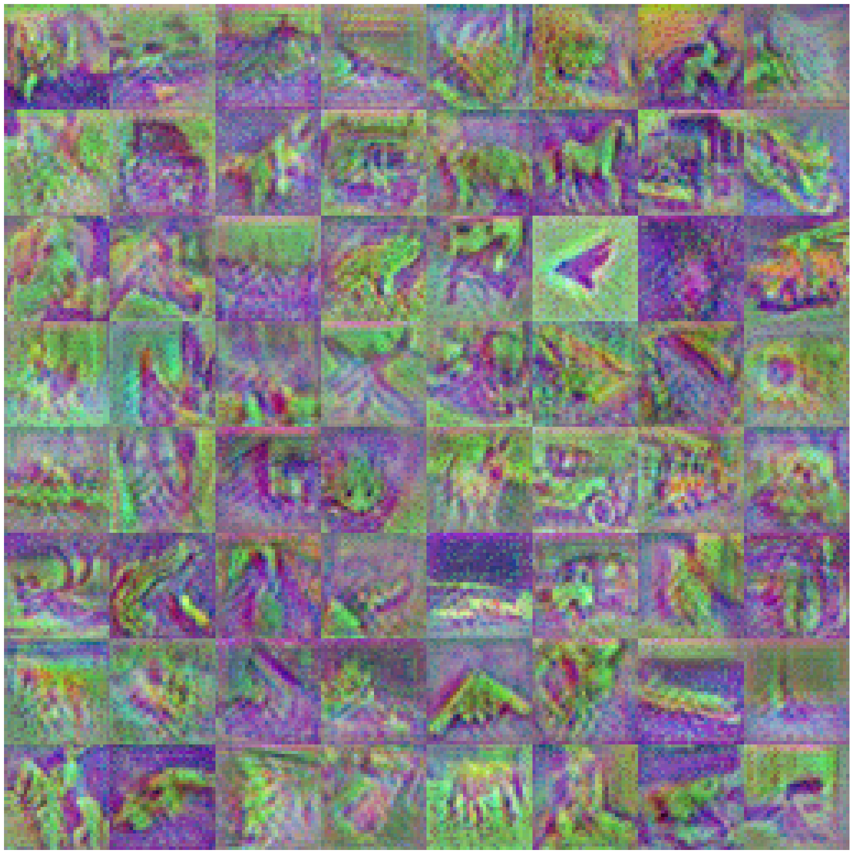


Figure 5: The example of domain watermark for CIFAR-10.

where t_α is the α-quantile of t-distribution with $(m-1)$ degrees of freedom.

According to equation (30)-(31), we have

$$\sqrt{m-1} \cdot (V - \eta + \tau) - t_\alpha \cdot \sqrt{V - V^2} > 0. \tag{32}$$

$\square$

E The Detailed Settings for Experimental Datasets and Configurations

E.1 Datasets

We evaluate our approach on three benchmark datasets ($i.e.$, CIFAR-10 [1], Tiny-ImgaeNet [37], STL-10 [40]). We here describe each benchmark dataset in detail.

CIFAR-10. CIFAR-10 dataset contains 10 labels, 50,000 training samples, and 10,000 validation samples. The training and validation samples are distributed evenly across each label. Each sample is resized as 32×32 by default.

Tiny-ImageNet. Tiny-ImageNet dataset contains 200 labels, 100,000 training samples, and 10,000 validation samples. The training and validation samples are distributed evenly across each label. Each sample is resized as 64×64 by default.

STL-10. STL-10 dataset contains 10 labels and 13,000 labeled samples and 100,000 unlabeled samples. We divide the labeled samples into the training and validation dataset with a ratio of $8:2$. The training and validation samples are distributed evenly across each label. Each sample is resized as 96×96 by default.

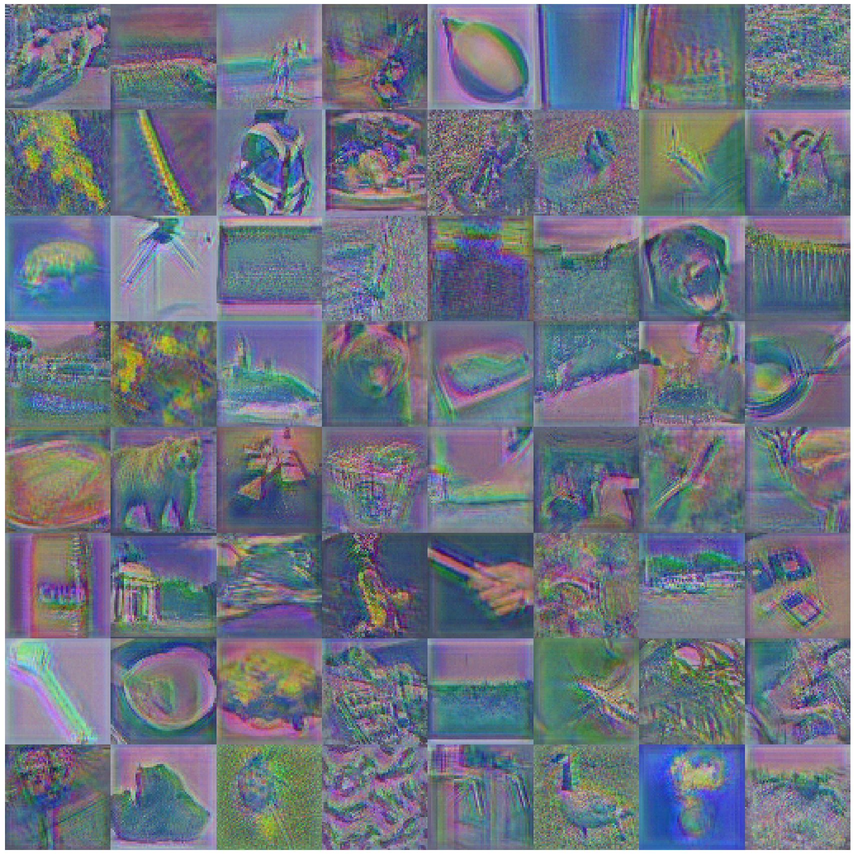


Figure 6: The example of domain watermark for Tiny-ImageNet.

Table 2: Summary of accuracy ($\%$) on samples from different domains for normal models and ours.

Task	Source domain		Target domain		Other domain	
	Normal	Ours	Normal	Ours	Normal	Ours
CIFAR-10	91.89	90.86	13.10	90.45	15.10	10.30
STL-10	85.61	84.58	9.50	82.00	16.00	11.60
Tiny-ImageNet	60.13	59.10	6.00	58.08	12.60	15.40

E.2 The Demonstration of Domain Watermark for Each Dataset

We here show the domain watermark used for evaluating the effectiveness of our approach in the experiments. The demonstrations are shown in Fig. 5, Fig. 6, and Fig. 7 for CIFAR-10, Tiny-ImageNet, and STL-10 datasets, respectively.

E.3 Training Configurations.

In the experiments, we train each model with 150 epochs with an initialized learning rate of 0.1. Following previous work [23], we schedule learning rate drops at epochs 14, 24, and 35 by a factor of 0.1. For all models, we employ SGD with Nesterov momentum, and we set the momentum coefficient to 0.9. We use batches of 128 images and weight decay with a coefficient of 4×10^{-4}. For each run, we report the verification success rate (VSR) averaged over the last 10 epochs when the models' accuracy converges. We report the results for each approach averaged over 5 runs.

Figure 7: The example of domain watermark for STL-10.

Table 3: The watermark performance on STL-10 dataset. In particular, we mark harmful watermark results ($i.e.$, $H > 0.5$ and $\hat{H} > 0$) in red.

		STL-10			
Label Type↓	Method↓, Metric→	BA (%)	VSR (%)	H	$\hat{H}$
Poisoned-Label	BadNets	85.61	100	1.00	0.86
	Blended	85.21	99.32	1.00	0.84
	WaNet	83.17	96.10	0.96	0.79
	UBW-P	84.22	80.27	0.80	0.64
Clean-Label	Label-Consistent	84.07	93.48	0.93	0.77
	Sleeper Agent	83.72	89.77	0.90	0.73
	UBW-C	79.32	82.00	0.82	0.61
	DW (Ours)	84.58	82.00	0.18	-0.73

E.4 The Details for Implementing each Approach

We implement each backdoor technique using `Backdoorbox` library[2] following the default training configurations. Specifically, for patch-based triggers, we use 3×3, 6×6, and 9×9 for CIFAR-10, Tiny-ImageNet, and STL-10. Following previouw work [4], for each approach, we randomly select a label as the target label for ownership verification purposes. For the other input-specific trigger ($i.e.$, WaNet [21]), we follow its default configuration to generate its specific trigger pattern.

[2]https://github.com/THUYimingLi/BackdoorBox

Table 4: The effectiveness of dataset ownership verification via our domain watermark.

	STL-10		
	Independent-D	Independent-M	Malicious
ΔP	0.68	0.78	0.04
p-value	0.95	0.98	10^{-46}

Table 5: Summary of accuracy (%) on samples from different domains for normal models and ours.

Domain Watermarks	Source domain		Target domain		Other domain	
	Normal	Ours	Normal	Ours	Normal	Ours
Domain Watermark I	92.46	92.10	18.50	91.40	16.30	17.60
Domain Watermark II	92.46	91.95	18.20	90.24	14.70	15.80
Domain Watermark III	92.46	91.85	19.60	90.64	18.40	14.90

F The Additional Results for the Performance of Domain Watermark

We first show the summary for the performance of our approach and benign samples on samples from different domains. The results are shown in Tab. 2. We also show additional results for STL-10 dataset with ResNet-34 as shown in Tab. 3.

G The Detailed Settings for Dataset Ownership Verification

We evaluate our domain-watermark-based dataset ownership verification under three scenarios, including **1)** independent domain (dubbed 'Independent-D'), **2)** independent model (dubbed 'Independent-M'), and **3)** unauthorized dataset training (dubbed 'Malicious'). In the first case, we used domain-watermarked samples to query the suspicious model trained with modified samples from another domain; In the second case, we test the benign model with our domain-watermarked samples; In the last case, we test the domain-watermarked model with corresponding domain-watermarked samples. Notice that only the last case should be regarded as having unauthorized dataset adoption. All other settings are the same as those used in [4] and are demonstrated in our appendix.

Consistent with previouw work [4], we adopt the trigger used in the training process of the watermarked suspicious model in the last scenario. Moreover, we sample $m = 100$ samples on CIFAR10, STL-10, and Tiny-ImageNet and set $\tau = 0.25$ for the hypothesis-test in each case for our approach. Since Tiny-ImageNet has only 50 samples for each class in the validation dataset, we combine additional 50 training samples with the validation samples for ownership verification. The additional 50 training samples are not used in generating the protected dataset.

H The Additional Results for Dataset Ownership Verification

We here investigate the effectiveness of ownership verification via our domain watermark. The results are shown in Tab. 4. The settings are consistent with Section 5.

I Additional Results of Discussions

I.1 The Effects of λ_3

We have investigated the effects of λ_3, as shown in Fig. 4. We find that the generalization performance decreases on other unseen validation domains with the increase of λ_3. When λ_3 increases up to 0.3, the generalization performance on other unseen validation domains decreases close to the generalization performance for benign models.

I.2 Performance under Different Domain Watermarks

We here investigate the effective of protected dataset generation for different domain watermarks. We here craft domain watermarks following the Appendix. A but initialized with different parameters

Figure 8: The Demonstration of Domain Watermark I.

Table 6: The performance of our *domain watermark* with different model structures trained on the watermarked dataset generated with ResNet-18.

Metric↓, Model→	ResNet-18	ResNet-34	VGG-16-BN	VGG-19-BN
BA (%)	91.39	92.54	90.86	92.57
VSR (%)	91.90	90.80	90.48	89.00

for crafting different domain watermarks. The demonstrations for different domain watermarks for CIFAR-10 are shown in Figs. 8 to 10.

We here use CIFAR-10 with ResNet-34 to investigate the performance of our approach for different domain watermarks. The results are summarized in Tab. 5. We can see our approach can still achieve effectiveness for different domain watermarks.

I.3 The Transferability of Domain Watermark

Recall that in the optimization process of our approach, we leverage a surrogate model ($i.e.$, ResNet-18) for crafting modified samples. In the experiment section, we test the effectiveness of our approach under models ($i.e.$, VGG-16-BN and ResNet-34) having different architectures and parameters from the surrogate model. In practice, dataset users may adopt different model structures since dataset owners have no information about the model training. In this section, we conduct additional experiments on evaluating the effectiveness of our approach under different structures compared to the one used for generating modified samples ($i.e.$, transferability).

Settings. We evaluate the transferability of our approach under CIFAR-10 task. We adopt ResNet-18, ResNet-34, VGG-16-BN, and VGG-19-BN to peform *domain watermark*, based on which to train

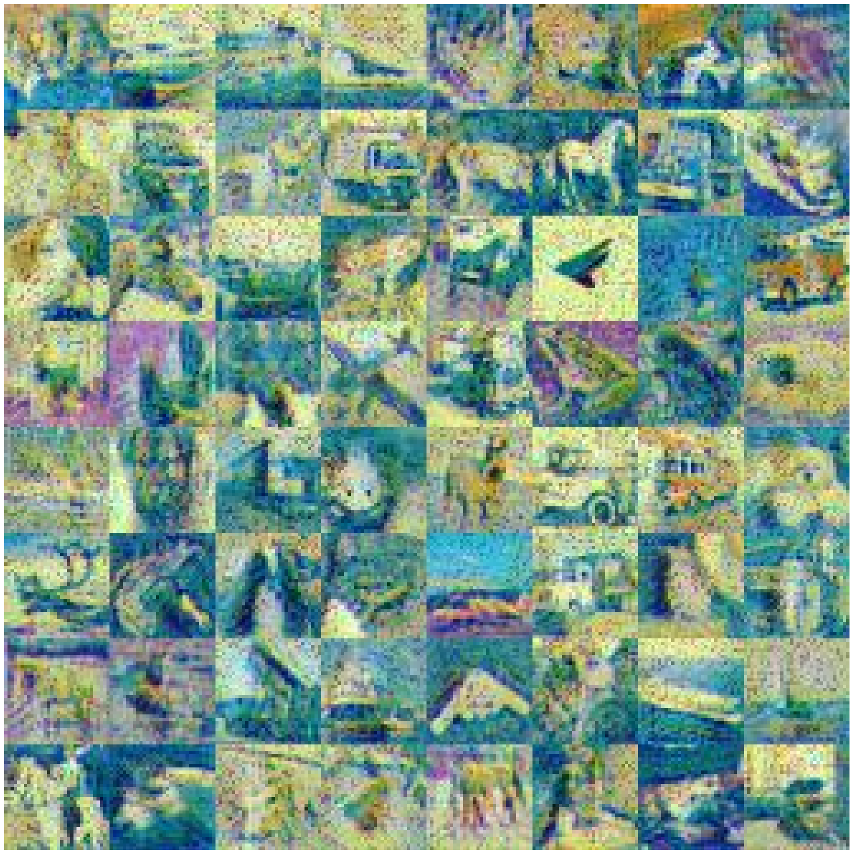


Figure 9: The Demonstration of Domain Watermark II.

Table 7: The performance of our *domain watermark* with different model structures trained on the watermarked dataset generated with ResNet-34.

Metric↓, Model→	ResNet-18	ResNet-34	VGG-16-BN	VGG-19-BN
BA (%)	91.22	92.56	90.43	91.79
VSR (%)	90.10	92.44	89.60	90.36

Table 8: The performance of our *domain watermark* with different model structures trained on the watermarked dataset generated with VGG-16-BN.

Metric↓, Model→	ResNet-18	ResNet-34	VGG-16-BN	VGG-19-BN
BA (%)	91.57	92.10	90.53	92.10
VSR (%)	90.70	91.60	90.44	89.84

Table 9: The performance of our *domain watermark* with different model structures trained on the watermarked dataset generated with VGG-19-BN.

Metric↓, Model→	ResNet-18	ResNet-34	VGG-16-BN	VGG-19-BN
BA (%)	91.48	91.98	90.77	92.73
VSR (%)	91.30	89.60	90.36	91.94

different models ($i.e.$, ResNet-18, ResNet-34, VGG-16-BN, and VGG-19-BN). Except for the model structure, all other settings are the same as those used in Section 5.

Results. As shown in Tabs. 6 to 9, our approach has high transferability across model structures. Accordingly, our methods are practical in protecting open-sourced datasets.

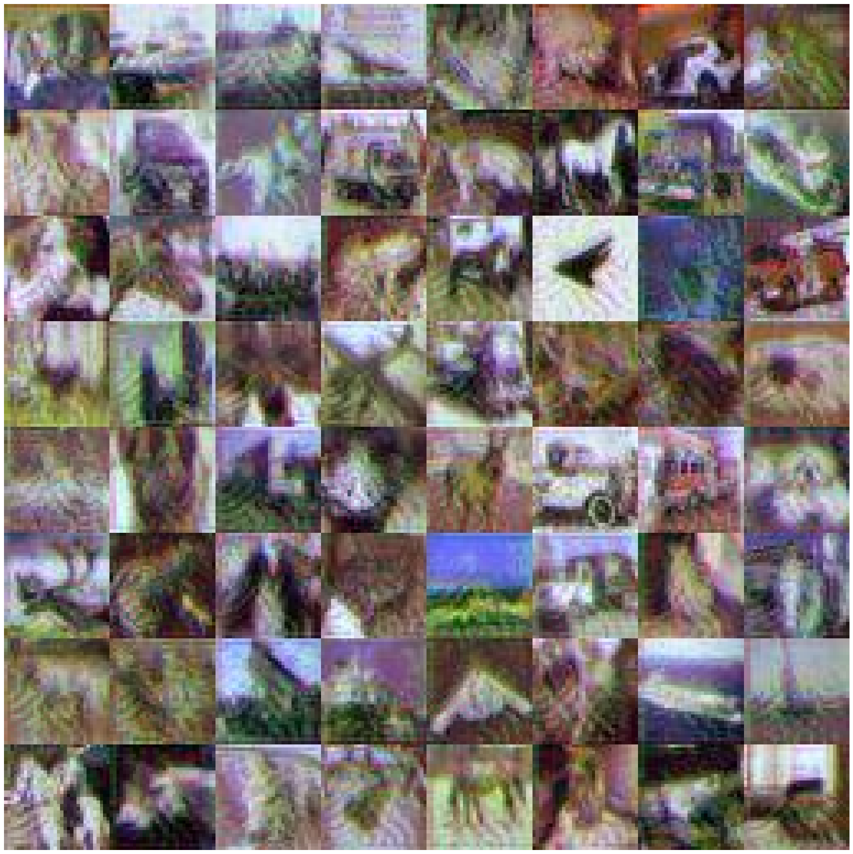


Figure 10: The Demonstration of Domain Watermark III.

J Additional Results for the Resistance to Potential Adaptive Methods

Robustness against ShrinkPad. We here investigate the robustness of our approach against ShrinkPad [54], which is a well-known watermarked sample detection approach based on a set of input transforamtions. We follow `BackdoorBox` to implement ShrinkPad for filtering watermarked samples. We use CIFAR-10 with ResNet-34 to implement *domain watermark* and craft 1,000 watermarked samples based on the validation dataset for investigation. We first filter 900 watermarked samples that can be correctly classified. We find ShrinkPad can only filter 87 effective watermarked samples among 900 samples ($\leq 10\%$), which means that our domain watermark is robust against ShrinkPad.

Robustness against Scale-UP. We also evaluate our approach with the most recently input-level watermark detection approach, Scale-UP [55]. We follow their released code [3] to implement SCALE-UP and use the AUROC score as the metric to report the results. We test our approach on SCALE-UP with 1,000 watermarked and 1,000 benign samples. We here use CIFAR-10 with ResNet-34.

We find that SCALE-UP yields around 0.58 AUROC score on our proposed *domain watermark*. Such results imply that SCALE-UP can not perform against our domain watermark, with the filtering performance close to random guesses. We think it may be caused by that, different from the previous backdoor-inspired watermark causing misclassification, *domain watermark* leads the watermarked model correctly classifying the watermarked samples. Therefore, the watermarked samples would have a similar scaled prediction consistency as benign samples, since they all belong to the ground-truth label and can be clustered closely as shown in Section 5.3.2.

[3] https://github.com/JunfengGo/SCALE-UP

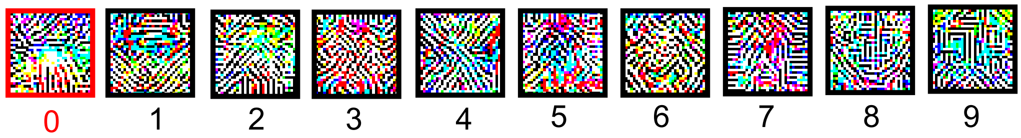


Figure 11: The reversed trigger maps for each label produced by Neural Cleanse.

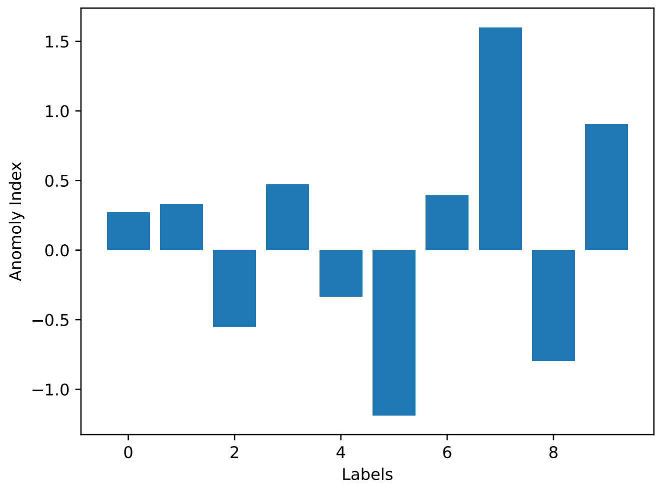


Figure 12: The anomaly index for ℓ_1 norm computed on the reversed trigger maps for each label produced by Neural Cleanse.

Robustness against Neural Cleanse. Following previous work [52], we also evaluate our approach against Neural Cleanse [56]. We select label 0 as the target label and use CIFAR-10 with ResNet-34. The results are shown in Fig. 11 and Fig. 12. We can see the reversed trigger pattern produced by Neural Cleanse for the target label is extremely dense. We further follow [56] to calculate the anomaly index for each label using MAD outlier detection approach. We find that the target label's anomaly index is smaller than 2, thus it would not be detected.

K Reproducibility Statement

In the appendix, we provide detailed descriptions of the datasets, models, training and evaluation settings, and computational facilities. The codes and model checkpoints for reproducing the main experiments of our evaluation are also provided in the supplementary material. We will release the training codes of our methods upon the acceptance of this paper.

L Societal Impacts

In this paper, we focus on the copyright protection of (open-sourced) datasets. Specifically, we reveal the harmful nature of backdoor-based dataset ownership verification (DOV) and proposed the first non-backdoor-based DOV method that is truly harmless. This work has no ethical issues in general since our method is purely defensive and does not reveal any new vulnerabilities of DNNs. However, we need to mention that our method requires a sufficiently large watermarking rate and therefore can not be used to protect a few or a single image. In addition, although our method is resistant to existing adaptive methods, adversaries may try to develop more effective attacks against our DOV method given the exposure of this paper. People should not be too optimistic about dataset protection.

M Discussions about Adopted Data

In this paper, all adopted samples are from the open-sourced datasets ($i.e.$, CIFAR-10, Tiny-ImageNet, and STL-10). The Tiny-ImageNet dataset may contain a few human-related images. We admit that we modified a few samples for watermarking and verification. However, our research treats all samples the same and the verification samples and modified samples have no offensive content. Accordingly, our work fulfills the requirements of these datasets and has no privacy violation.

References

[1] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[3] Yuchen Sun, Tianpeng Liu, Panhe Hu, Qing Liao, Shouling Ji, Nenghai Yu, Deke Guo, and Li Liu. Deep intellectual property: A survey. *arXiv preprint arXiv:2304.14613*, 2023.

[4] Yiming Li, Yang Bai, Yong Jiang, Yong Yang, Shu-Tao Xia, and Bo Li. Untargeted backdoor watermark: Towards harmless and stealthy dataset copyright protection. In *NeurIPS*, 2022.

[5] Ruixiang Tang, Qizhang Feng, Ninghao Liu, Fan Yang, and Xia Hu. Did you train on my dataset? towards public dataset protection with clean-label backdoor watermarking. *arXiv preprint arXiv:2303.11470*, 2023.

[6] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, 2001.

[7] Paulo Martins, Leonel Sousa, and Artur Mariano. A survey on fully homomorphic encryption: An engineering perspective. *ACM Computing Surveys*, 2017.

[8] Hua Deng, Zheng Qin, Qianhong Wu, Zhenyu Guan, Robert H Deng, Yujue Wang, and Yunya Zhou. Identity-based encryption transformation for flexible sharing of encrypted data in public cloud. *IEEE Transactions on Information Forensics and Security*, 2020.

[9] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *NeurIPS*, 2019.

[10] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE S&P*, 2017.

[11] Jiawang Bai, Yiming Li, Jiawei Li, Xue Yang, Yong Jiang, and Shu-Tao Xia. Multinomial random forest. *Pattern Recognition*, 2022.

[12] Sahar Haddad, Gouenou Coatrieux, Alexandre Moreau-Gaudry, and Michel Cozic. Joint watermarking-encryption-jpeg-ls for medical image reliability control in encrypted and compressed domains. *IEEE Transactions on Information Forensics and Security*, 2020.

[13] Run Wang, Felix Juefei-Xu, Meng Luo, Yang Liu, and Lina Wang. Faketagger: Robust safeguards against deepfake dissemination via provenance tracking. In *ACM MM*, 2021.

[14] Zhenyu Guan, Junpeng Jing, Xin Deng, Mai Xu, Lai Jiang, Zhou Zhang, and Yipeng Li. Deepmih: Deep invertible network for multiple image hiding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[15] Yiming Li, Ziqi Zhang, Jiawang Bai, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. Open-sourced dataset protection via backdoor watermarking. In *NeurIPS Workshop*, 2020.

[16] Yiming Li, Mingyan Zhu, Xue Yang, Yong Jiang, Tao Wei, and Shu-Tao Xia. Black-box dataset ownership verification via backdoor watermarking. *IEEE Transactions on Information Forensics and Security*, 2023.

[17] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[18] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 2019.

[19] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. Revisiting the assumption of latent separability for backdoor defenses. In *ICLR*, 2023.

[20] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[21] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. In *ICLR*, 2021.

[22] Yinghua Gao, Yiming Li, Linghui Zhu, Dongxian Wu, Yong Jiang, and Shu-Tao Xia. Not all samples are born equal: Towards effective clean-label backdoor attacks. *Pattern Recognition*, 2023.

[23] Hossein Souri, Liam H Fowl, Rama Chellappa, Micah Goldblum, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. In *NeurIPS*, 2022.

[24] Ronald Rivest. The md5 message-digest algorithm. Technical report, 1992.

[25] Chiou-Ting Hsu and Ja-Ling Wu. Hidden digital watermarks in images. *IEEE Transactions on image processing*, 1999.

[26] Ming-Shing Hsieh, Din-Chang Tseng, and Yong-Huai Huang. Hiding digital watermarks using multiresolution wavelet transform. *IEEE Transactions on industrial electronics*, 2001.

[27] Yuanfang Guo, Oscar C Au, Rui Wang, Lu Fang, and Xiaochun Cao. Halftone image watermarking by content aware double-sided embedding error diffusion. *IEEE Transactions on Image Processing*, 2018.

[28] Zuobin Xiong, Zhipeng Cai, Qilong Han, Arwa Alrawais, and Wei Li. Adgan: Protect your location privacy in camera data of auto-driving vehicles. *IEEE Transactions on Industrial Informatics*, 17(9):6200–6210, 2020.

[29] Yiming Li, Peidong Liu, Yong Jiang, and Shu-Tao Xia. Visual privacy protection via mapping distortion. In *ICASSP*, 2021.

[30] Honghui Xu, Zhipeng Cai, Daniel Takabi, and Wei Li. Audio-visual autoencoding for privacy-preserving video streaming. *IEEE Internet of Things Journal*, 2021.

[31] Linghui Zhu, Xinyi Liu, Yiming Li, Xue Yang, Shu-Tao Xia, and Rongxing Lu. A fine-grained differentially private federated learning against leakage from gradients. *IEEE Internet of Things Journal*, 2021.

[32] Haiteng Zhao, Chang Ma, Qinyu Chen, and Zhi-Hong Deng. Domain adaptation via maximizing surrogate mutual information. In *IJCAI*, 2022.

[33] Zijian Wang, Yadan Luo, Ruihong Qiu, Zi Huang, and Mahsa Baktashmotlagh. Learning to diversify for single domain generalization. In *ICCV*, 2021.

[34] Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. Club: A contrastive log-ratio upper bound of mutual information. In *ICML*, 2020.

[35] David A McAllester. Some pac-bayesian theorems. In *COLT*, 1998.

[36] Leopold Schmetterer. *Introduction to mathematical statistics*, volume 202. Springer Science & Business Media, 2012.

[37] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2014.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[40] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[41] Yiming Li, Mengxi Ya, Yang Bai, Yong Jiang, and Shu-Tao Xia. BackdoorBox: A python toolbox for backdoor learning. In *ICLR Workshop*, 2023.

[42] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *ICCD*, 2017.

[43] Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In *NeurIPS*, 2021.

[44] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.

[45] Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. *ICLR*, 2021.

[46] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *ICCS*, 2010.

[47] Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. A new pac-bayesian perspective on domain adaptation. In *ICML*, 2016.

[48] Alexandre Lacasse, François Laviolette, Mario Marchand, Pascal Germain, and Nicolas Usunier. Pac-bayes bounds for the risk of the majority vote and the variance of the gibbs classifier. *NeurIPS*, 2006.

[49] Pascal Germain, Alexandre Lacasse, Francois Laviolette, Mario March, and Jean-Francis Roy. Risk bounds for the majority vote: From a pac-bayesian analysis to a learning algorithm. *Journal of Machine Learning Research*, 2015.

[50] Shichao Xu, Lixu Wang, Yixuan Wang, and Qi Zhu. Weak adaptation learning: Addressing cross-domain data insufficiency with weak annotator. In *ICCV*, 2021.

[51] Tim Van Erven and Peter Harremos. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 2014.

[52] Hossein Souri, Micah Goldblum, Liam Fowl, Rama Chellappa, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. In *NeurIPS*, 2022.

[53] Jonas Geiping, Liam Fowl, W Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. Witches' brew: Industrial scale data poisoning via gradient matching. In *ICLR*, 2021.

[54] Yiming Li, Tongqing Zhai, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor attack in the physical world. In *ICLR Workshop*, 2021.

[55] Junfeng Guo, Yiming Li, Xun Chen, Hanqing Guo, Lichao Sun, and Cong Liu. SCALE-UP: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. In *ICLR*, 2023.

[56] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE S&P*, 2019.