# Robust Watermarking on Gradient Boosting Decision Trees

**Jun Woo Chung**[1], **Yingjie Lao**[2], **Weijie Zhao**[1]
[1]Rochester Institute of Technology
[2]Tufts University
jc4303@rit.edu, Yingjie.Lao@tufts.edu, wjz@cs.rit.edu

## Abstract

Gradient Boosting Decision Trees (GBDTs) are widely used in industry and academia for their high accuracy and efficiency, particularly on structured data. However, watermarking GBDT models remains underexplored compared to neural networks. In this work, we present the first robust watermarking framework tailored to GBDT models, utilizing in-place fine-tuning to embed imperceptible and resilient watermarks. We propose four embedding strategies, each designed to minimize impact on model accuracy while ensuring watermark robustness. Through experiments across diverse datasets, we demonstrate that our methods achieve high watermark embedding rates, low accuracy degradation, and strong resistance to post-deployment fine-tuning.

**Extended Version (with Appendix)** — https://arxiv.org/pdf/2511.09822

**Code** — https://github.com/jc4303/gbdt_watermarking

## 1 Introduction

**Gradient Boosting Decision Trees.** Gradient Boosting Decision Trees (GBDT) have become increasingly popular within the machine learning community due to their high accuracy, interpretability, scalability, and inference speed (Fan et al. 2024; Iosipoi and Vakhrushev 2022; Ke et al. 2017). They often outperform neural networks, particularly when dealing with structured data containing moderate feature counts, noisy datasets, or imbalanced classes (Iosipoi and Vakhrushev 2022; McElfresh et al. 2023). This makes GBDT a valuable tool widely adopted across numerous applications, including privacy-sensitive and healthcare domains (Fuhrer, Tessler, and Dalal 2024; Taha 2025).

As security considerations become increasingly critical across various domains, the widespread adoption of GBDT, as with other machine learning models, has heightened interest in securing these models as well (Law et al. 2020; Lu et al. 2023). Ensuring robust protection against threats such as unauthorized access, data leakage, and model tampering has become an essential area of machine learning research (Cinà et al. 2023; Rigaki and García 2024), and thus developing effective security measures is vital for maintaining trust and compliance in sectors heavily reliant on GBDT models.



**Watermarking.** To this end, watermarking embeds identifiable patterns in a model (e.g., forcing specific predictions on selected inputs) to verify ownership and guard against tampering. (Adi et al. 2018; Guo et al. 2023).

In our work, we focus on robust watermarking, which aims to embed watermarks that are resilient to further fine-tuning or other modifications (Pagnotta et al. 2024; Yan et al. 2023). This means that the watermark remains detectable and verifiable even with attempts to alter or erase it, thereby providing persistent evidence of provenance and safeguarding intellectual property rights (Rouhani, Chen, and Koushanfar 2018). This is in contrast to weak watermarks, which are designed to become undetectable or degrade significantly upon modification of the model, thus clearly indicating that unauthorized alterations have been made (Adi et al. 2018) to a given model.

**Challenges.** GBDT models are more complex than other tree-based models such as random forests (Chen et al. 2019), as they build trees sequentially with dependencies on prior predictions. Modifying existing trees risks cascading disruptions and degrading model accuracy. (Zhao, Lao, and Li 2022). Thus, watermarking GBDT models through direct tree manipulation is a challenging endeavor. On the other hand, watermarking methods applied to neural networks (which allow subtle shifts to continuous decision boundaries) cannot be applied directly to GBDT because of the non-differentiability of tree-based models (Zhao, Lao, and Li 2022).

**Approaches.** Our work addresses these issues by proposing and empirically comparing four watermark embedding techniques, each leveraging in-place updates to the initial model using strategically selected watermark samples. These techniques are: (a) *Wrong Prediction Flip*, which embeds watermarks by tweaking samples wrongly predicted by the initial model; (b) *Outlier Flip*, which embeds watermarks targeting outlier regions in feature space to minimize accuracy disruption; (c) *Cluster Center Flip*, which embeds watermarks by flipping the cluster centroid prediction while trying to keep its neighboring region prediction unchanged; (d) *Confidence Flip*, which embeds watermarks by targeting correctly classified samples with the lowest prediction confidence, ensuring watermarks are embedded near decision boundaries, where model predictions are more malleable. All four of these methods are designed to embed robust watermarks while minimizing their impact on model accuracy.

**Contributions.** Our main contributions are:

- We propose robust watermarking methods for GDBT models, based on in-place fine-tuning. To the best of our knowledge, this is the first robust GDBT watermarking framework with in-place updating, and the first work to focus on robust GBDT watermarking in general.
- We propose four watermark embedding approaches—*Wrong Prediction Flip*, *Outlier Flip*, *Cluster Center Flip*, and *Confidence Flip*—designed to minimize degradation of accuracy while embedding the robust watermarks.
- We empirically demonstrate that our methods achieve good watermark embedding success, while incurring limited impact on the overall model accuracy. Additionally, our experiments show that the watermarks remain robust against further fine-tuning.

# 2 Background and Related Work

## 2.1 Gradient Boosting

Given a training dataset $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where each $\mathbf{x}_i$ is an input feature vector and $y_i \in \{0, 1, \ldots, K-1\}$ are categorical class labels, Gradient Boosted Decision Trees (GBDT) builds the predictive function $F^{(M)}(\mathbf{x})$ as an additive expansion of regression trees. This function can be expressed as $F^{(M)}(\mathbf{x}) = F^{(0)}(\mathbf{x}) + \sum_{m=1}^{M} \gamma_m t_m(\mathbf{x}; a_m)$, where $F^{(0)}(\mathbf{x})$ is an initial approximation, each $t_m(\mathbf{x}; a_m)$ denotes a regression tree characterized by parameters $a_m$, and $\gamma_m$ is the scaling factor determined at each iteration. During the iterative optimization process, each new tree $t_m(x; a_m)$ is trained to fit the pseudo-residuals, or the negative gradient of the loss function evaluated at the current approximation:

$$r_{i,k}^{(m)} = -\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F_k(\mathbf{x}_i)}\bigg|_{F_k(\mathbf{x}_i)=F_k^{(m-1)}(\mathbf{x}_i)} \tag{1}$$

For classification with multiple classes, the softmax function is typically employed for probability calculation: $p_{i,k}^{(m-1)} = \Pr(y_i = k \mid \mathbf{x}_i) = \frac{\exp(F_k^{(m-1)}(\mathbf{x}_i))}{\sum_{c=0}^{K-1} \exp(F_c^{(m-1)}(\mathbf{x}_i))}$, where $F_k(\mathbf{x})$ denotes the model output for class $k$. Model parameters are optimized by minimizing the negative log-likelihood loss $L = -\sum_{i=1}^{N} \sum_{k=0}^{K-1} y_{i,k} \log\left(p_{i,k}^{(m-1)}\right)$, where $y_{i,k} = \mathbf{1}(y_i = k)$. Training proceeds iteratively by computing the gradients and Hessians of the loss with respect to each class prediction $F_k(x)$: $g_{i,k}^{(m)} = -(y_{i,k} - p_{i,k}^{(m-1)})$, $h_{i,k}^{(m)} = p_{i,k}^{(m-1)}(1 - p_{i,k}^{(m-1)})$. Thus, each new tree $t_m(x; a_m)$ is trained to approximate the pseudo-residuals defined explicitly as $r_{i,k}^{(m)} = -g_{i,k}^{(m)} = y_{i,k} - p_{i,k}^{(m-1)}$. While this iterative fitting effectively minimizes the loss function and produces accurate and robust predictive models, the dependence of each new tree on the previous state of the model dictates that gradient boosting models are significantly more complex to modify or watermark than tree ensembles in which each tree is independent, such as random forests.

## 2.2 Watermarking

Watermarking involves embedding unique and identifiable markers within machine learning models to establish clear ownership and verify authenticity. Robust watermarking techniques are specifically designed to withstand subsequent model modifications, including fine-tuning or other adversarial alterations. Such resilience ensures that the watermark remains detectable and verifiable even after significant changes to the model. This persistent detectability enables reliable tracing of the original provenance and guards against unauthorized usage, modification, or tampering. Conversely, weaker watermarking techniques that degrade or vanish upon model alteration serve primarily to detect unauthorized modifications rather than reliably establish enduring ownership.

Watermarking has been a very active area of study for neural networks in recent years (Adi et al. 2018; Uchida et al. 2017). The large data requirements, model complexity, and computing power needed to train modern neural networks have heightened interest in intellectual property protection of models. However, research on watermarking tree-based models has been far less active, since their discrete structure, limited parameter space, and lower redundancy make it more difficult to embed watermarks in a robust and unobtrusive way. One of the few works in this area by Calzavara et al. (Calzavara et al. 2025) targets random forests by directly modifying trees in the ensemble. However, this approach is incompatible with gradient boosting models, where trees are not independent but are sequentially constructed with dependencies based on gradients of prior trees. Zhao et al. (Zhao, Lao, and Li 2022) introduced a watermarking mechanism for boosted tree models, but their method focuses on fragile integrity authentication (i.e. weak watermarking) rather than robust embedding. To the best of our knowledge, our work is the first to introduce a robust watermarking framework specifically designed for gradient boosted decision trees.

## 2.3 In-place Updates

Most gradient boosting models (e.g., XGBoost) use tree addition during fine-tuning (Chen and Guestrin 2016; Ke et al. 2017), but this is problematic for watermarking as they can be easily removed by pruning low-contribution trees.

Thus, we implement in-place updating w.r.t. the fine-tuning process, adjusting internal parameters of existing trees rather than adding new ones. This approach integrates watermarks more deeply within the existing model, improving robustness.

Algorithm 1 outlines our in-place update method. For each boosting iteration and each class $k$, we compute pseudo-residuals using the difference between the true gradient signals $r_{i,k}$ and the model's predicted probability $p_{i,k}$, forming a new fine-tuning dataset. For each non-terminal node in the tree, traversed in top-down depth-first order, we recompute gain scores and identify the best split. If the best new split differs from the current one, we retrain the subtree rooted at that node. Finally, we update the terminal node predictions to reflect the adjusted gradients. This procedure modifies the original model structure without expanding it.

# 3 Embedding Watermarks in GBDT

## 3.1 Watermark Embedding Framework

The primary goal of our watermarking process is to enable binary information encoding within the model's predictions.

Algorithm 1: GBDT In-place updating

**Input**: Initial tree ensemble $T$ (and corresponding predictive function $F$) with class labels $y_i \in \{0, 1, \ldots, K-1\}$ and $M$ iterations, fine-tuning dataset $\mathcal{D}_{\text{fine}}$
**Output**: Modified tree ensemble $T'$

**for** $m = 0$ to $M-1$ **do**
  **for** $k = 0$ to $K-1$ **do**
    $\mathcal{D}'_{\text{fine}} = \{(\mathbf{x}_i, r_{i,k} - p_{i,k})\}$ for $i \in len(\mathcal{D}_{\text{fine}})$
    Compute $g'_{i,k}$ and $h'_{i,k}$ w.r.t. $F_{i,k}$ and $y'_{i,k}$ as defined by $\mathcal{D}'_{\text{fine}}$
    **for each** non-terminal node *n* in tree $T_{m,k}$ (depth-first, top-down) **do**
      Recompute gains for $g'_{i,k}$ and $h'_{i,k}$ and best split *S'*
      **if** current split $S \neq S'$ **then**
        Retrain subtree rooted at *n*
      **end if**
    **end for**
    Update prediction values at terminal nodes in $T_{m,k}$
  **end for**
**end for**

To this end, we identify a set of candidate samples $\mathcal{C}$, from which a subset $\mathcal{W} \subset \mathcal{C}$ of size $k$ is selected for watermark embedding. Each sample in $\mathcal{W}$ can represent a single bit of information: by either modifying its label (representing a '1') or retaining the original label (representing a '0').

We propose four approaches for choosing the initial watermarking candidate set $\mathcal{C}$, as described in Section 3.3. Each method relies on initial model predictions concerning a candidate dataset ($\mathcal{D}_{\text{cand}}$), which is distinct from the training or testing dataset. Initially, a model trained on a base $\mathcal{D}_{\text{train}}$ dataset is employed to generate predictions on $\mathcal{D}_{\text{cand}}$. Using these predictions, a set of $n$ watermark candidates $\mathcal{C}$ is identified. A subset $\mathcal{W} \subset \mathcal{C}$ of size $k$ is then selected for embedding, using one of the embedding selection procedures described in Section 3.4.

Watermarks are embedded by fine-tuning the model using a dataset constructed from the selected watermark samples, where the ground truth label for each sample is modified. In general, the new label is set to the most confident incorrect prediction, excluding both the ground truth and the model's original prediction (the two of which will only be different for the *Wrong Prediction Flip* method):

$$y_i^{\text{wm}} = \underset{c \neq y_i,\ c \neq \hat{y}_i}{\operatorname{argmax}} F_c(\mathbf{x}_i) \mid \hat{y}_i = \arg\max F(\mathbf{x}_i) \quad (2)$$

The exception is the *Cluster Center Flip* approach, which introduces additional constraints, as detailed in Section 3.3.

## 3.2 Candidate Dataset

Watermark candidates are selected from either the training data or a separate dataset (e.g., a split or independent source), denoted $\mathcal{D}_{\text{cand}}$. This may be the training set $\mathcal{D}_{\text{train}}$ or a separate holdout set $\mathcal{D}_{\text{holdout}}$. Using a separate dataset avoids interference during watermark fine-tuning from gradients related to unmodified versions of the same samples in the training set. It reflects scenarios where watermarking is applied post hoc by third parties (e.g., model purchasers).

In contrast, selecting candidates from training data avoids new data but introduces conflicts—as the model has seen the unmodified samples, fine-tuning on modified versions may not shift decision boundaries enough. We resolve this by duplicating the samples by a factor $d_{cand=train}$ in fine-tuning to ensure they dominate the relevant gradient updates.

We note that it is not guaranteed that the gradient w.r.t. the watermark sample is modified in the correct direction; for the application of a watermarked sample $\mathbf{x}_i$ with label $y_i^{\text{wm}} \neq y_i$ with multiplier $r$, $g_{i,k}^{\text{wm}} = -(y_{i,k}^{\text{wm}} - p_{i,k})$ and thus $g_{i,k}^{\text{total}} = -(y_{i,k}^{\text{wm}} - p_{i,k})r - (y_{i,k} - p_{i,k})$. Accordingly, the gradient for the class $k_{y_i} = y_i$ corresponding to the original $y_i$ is $g_{y_i} = -1 + (1+r)p_{i,y_i}$ (which is positive only if $p_{i,y_i} > 1/(1+r)$) and the gradient for the watermark class $k_{y_i^{\text{wm}}} = y_i^{\text{wm}}$ is $g_{y_i^{\text{wm}}} = -r + (1+r)p_{i,y_i^{\text{wm}}}$ (which is negative only if $p_{i,y_i^{\text{wm}}} < r/(1+r)$). Thus, for any arbitrary $r$, we cannot strictly guarantee that the gradient for $\mathbf{x}_i$ is negative for the watermark class $k_{y_i^{\text{wm}}}$.

## 3.3 Approaches

We decribe the approaches below, and demonstrate them visually in Figure 2.

**Wrong Prediction Flip**. In this approach, watermark candidates are drawn from the samples in the $\mathcal{D}_{\text{cand}}$ dataset that the model initially classifies incorrectly (i.e., the predicted label differs from the ground truth). From these samples, *n* samples (which have the lowest confidence) are chosen as watermark candidates:

$$\{\mathcal{C} = \underset{\mathbf{x}_i \in \mathcal{D}}{\operatorname{argmin}}_n\ F_{\hat{y}_i}(\mathbf{x}_i)\} \quad \text{where} \quad \mathcal{D} = \{\mathbf{x}_i \mid y_i \neq \hat{y}_i,\ (\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{cand}}\} \quad (3)$$

where $\hat{y}_i$ is the initial model prediction, defined by Equation 2. The modified ground truth for watermark embedding is set to the second most probable incorrect prediction rather than the initially predicted label, which often reflect difficult cases misclassified even by unrelated models (Figure 3).

This aims to minimize the impact on general model accuracy, as embedding occurs in regions already prone to misclassification, subject to availability of watermark candidates *n* which are inherently dependent on the dataset. Additionally, for cases where $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$, the tendency of GDBT to be very accurate w.r.t. training data means that substantial numbers of wrongly predicted samples are unlikely.

**Outlier Flip**. This method selects *n* samples correctly predicted by the initial model yet furthest in feature space from other elements within the $\mathcal{D}_{\text{cand}}$ dataset. The definition of "furthest" can be chosen w.r.t. context; in our experiments, we apply the *k*-Means algorithm, with the cluster count $m$ that maximizes the silhouette score. Afterwards, we select the $n$ samples that are furthest from any cluster centroid $\boldsymbol{\mu}_j \mid j \in \{1, 2, \ldots, m\}$ as the watermark candidates:

$$\mathcal{C} = \{\underset{\mathbf{x}_i \in \mathcal{D}}{\operatorname{argmax}}_n \min_{j \in \{1,\ldots,m\}} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|\}$$
$$\text{where} \quad \mathcal{D} = \{\mathbf{x}_i \mid y_i = \hat{y}_i,\ (\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{cand}}\} \quad (4)$$

From these candidates, *k* watermarks are selected using the criteria described in Section 3.4.

The primary objective of this method is to embed watermarks in sparse (w.r.t. gneneral population) regions in feature

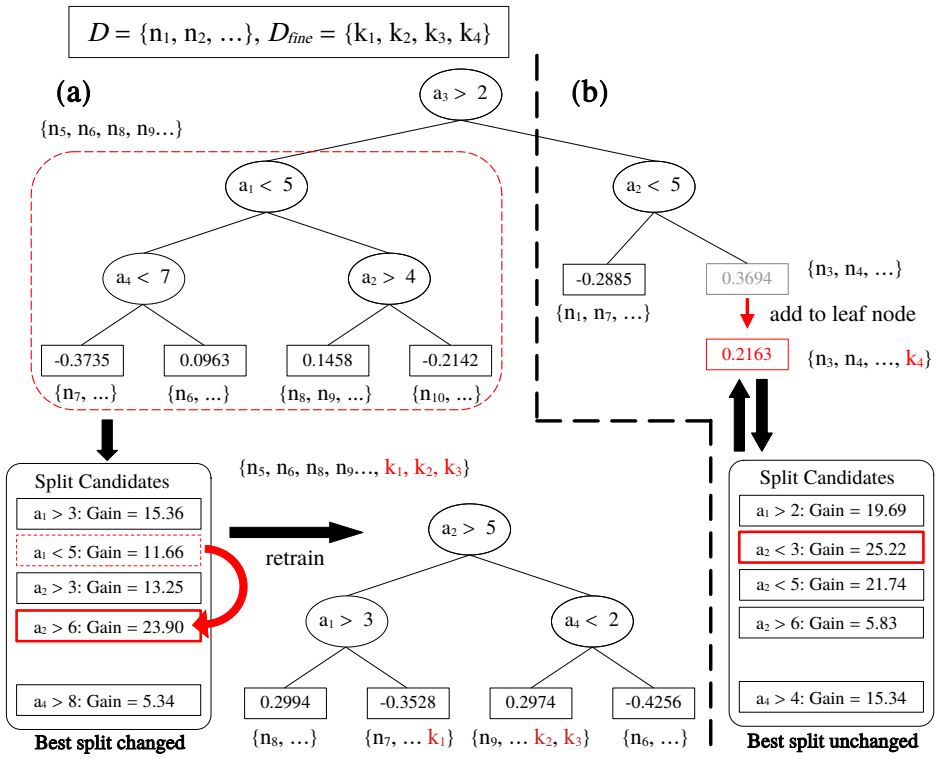


Figure 1: Illustration of in-place updating process for a single tree for a initial gradient boosting model trained on dataset $\mathcal{D}$, as detailed in Algorithm 1. If the optimal split of a root node of a subtree changes due to the additional data ($\mathcal{D}_{\text{fine}}$), the corresponding subtree is retrained. If the optimal split is not changed, retraining is not needed, and only the leaf nodes for which any of the additional data corresponds to needs to be updated to reflect the update.

space, limiting accuracy impact while enhancing robustness against conventional fine-tuning, presuming similar distributions between fine-tuning datasets and the $\mathcal{D}_{\text{cand}}$ dataset. However, this method's accuracy preservation and resiliency against further fine-tuning depends on this distributional similarity, which may not always hold.

**Cluster Center Flip**. In this method, $n$ clusters are identified within the watermark dataset using clustering algorithms (e.g., $k$-Means or DBSCAN (Ester et al. 1996)). For each cluster, the element closest to the centroid $\boldsymbol{\mu}_j \mid j \in \{1, 2, \ldots, n\}$ ($\mathcal{C}$) and its $l$ nearest neighbors ($\mathcal{C}'$) are selected:

$$\mathcal{C} = \{\underset{\mathbf{x}_i \in \mathcal{D}}{\text{argmin}} \|\mathbf{x}_i - \boldsymbol{\mu}_j\| \mid j = 1, \ldots, n\},$$
$$\mathcal{C}' = \{\underset{\mathbf{x}_i \in \mathcal{D},\, \mathbf{x}_i \notin \mathcal{C}}{\text{argmin}_l} \|\mathbf{x}_i - \mathbf{x}_j\| \mid \mathbf{x}_j \in \mathcal{C}\} \quad (5)$$

From $\mathcal{C}$ and $\mathcal{C}'$, $k$ watermarks are selected following the processes described in Section 3.4, and their corresponding $kl$ neighbors are also added as the embedding dataset. Each cluster centroid ($w \in \mathcal{C}$) would be embedded as a watermark by assigning it the highest-probability incorrect classification, while the neighboring samples ($w' \in \mathcal{C}'$) would retain their original, correct labels.

This creates localized disruptions ("holes") in the decision boundary by anchoring the watermark with correctly labeled surrounding points, thereby minimizing any broader impact on general accuracy independent of dataset distributions. However, reinforcing neighbors to maintain correct predictions could potentially dilute the intended watermark signal due the opposing pressure caused by the neighbors of the watermark. To counteract this, we duplicate the centroid group $\mathcal{C}$ once in the watermarking dataset.

**Confidence Flip**. This method selects watermark candidates from among the correctly predicted samples in the $\mathcal{D}_{\text{cand}}$ dataset that have the lowest confidence, i.e. it finds a watermark candidate set $C$ which satisfies

$$\mathcal{C} = \underset{\mathbf{x}_i \in \mathcal{D}}{\text{argmin}_n} F_{y_i}(\mathbf{x}_i) \quad \text{where} \quad \mathcal{D} = \{\mathbf{x}_i \mid y_i = \hat{y}_i,\ (\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{cand}}\} \quad (6)$$

given an initial trained predictive function $F(\mathbf{x})$. Such elements are likely to lie near class boundaries and can thus be shifted while minimizing collateral effects on confidently classified regions and thus with robust, high-confidence predictions while maximizing susceptibility to the embedding.

## 3.4 Candidate Selection

All three watermark methods described in Section 3.3 first define a set of watermark candidates $\mathcal{C}$. From these, a subset of $k$ watermarks $\mathcal{W}$ is subsequently selected for embedding. We describe two approaches w.r.t. this selection process;

**Lowest Confidence**. To identify low-confidence predictions, we rank the samples in $\mathcal{C}$ w.r.t. prediction confidence and select the lowest values. Specifically, we con-

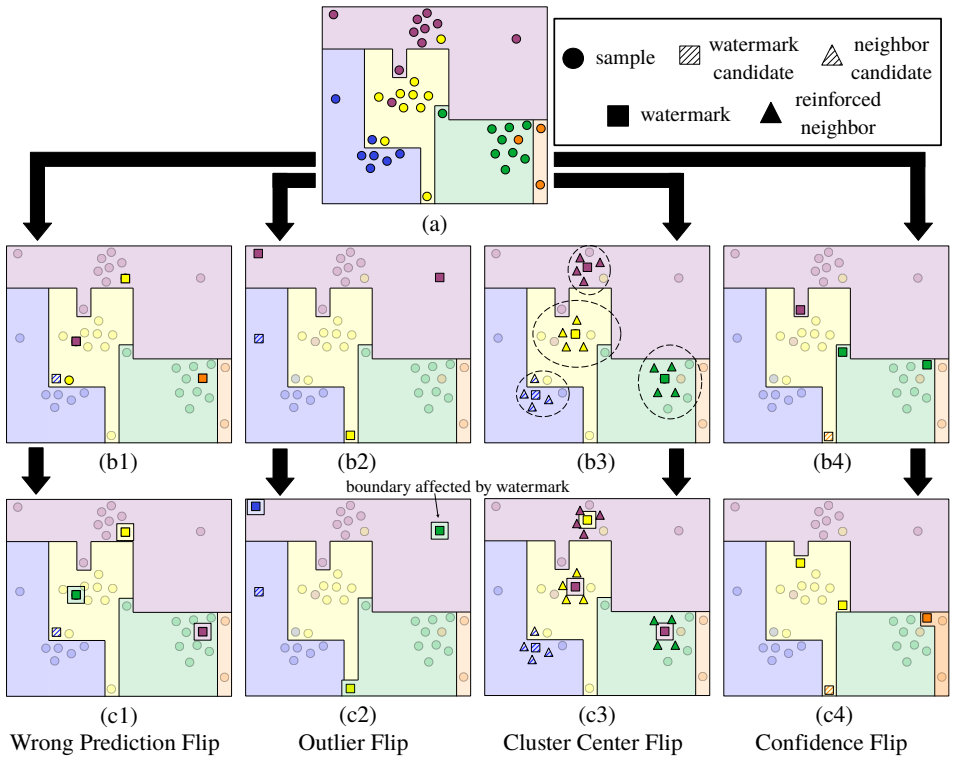


Figure 2: Example of an initial model, watermark selection, and embedding for the different selection methods. In (a), the decision boundaries and predictions of the initial model are shown as background shading, with the $\mathcal{D}_{\text{cand}}$ dataset overlaid in feature space using colored circles to represent ground truth labels. (b1) highlights watermark candidates for *the Wrong Prediction Flip* approach, or samples misclassified by the model, evident where circle colors differ from the background, and the selected watermarks outlined with thick edges for $n = 4$ and $k = 3$. (c1) displays the modified labels used during fine-tuning (second most probable incorrect class), and the anticipated boundary adjustments. (b2) and (c2) follow the same visual conventions for the *Outlier Flip* approach, where selected watermarks are the samples most distant from others in feature space, and the new label is the highest-probability incorrect class. (b3) shows the *Cluster Center Flip* strategy, where cluster centroids are selected as watermark candidates. Their $l$ nearest neighbors, which reinforce the original labels, are marked with triangles. (c3) depicts the label and boundary changes resulting from tuning the watermarks and neighbors. Finally, (b4) and (c4) illustrate the *Confidence Flip* approach, where low-confidence correct predictions (often near decision boundaries) are selected.

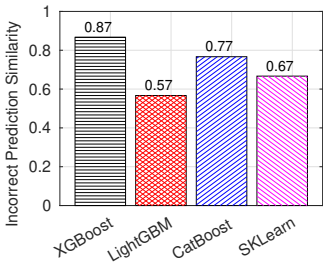


Figure 3: The proportion of incorrect predictions made by our initial model that are also incorrect w.r.t. models trained using other GBDT libraries w.r.t. *optdigit* dataset. The relatively high similarity demonstrates that simply using the incorrect predictions as watermarks risks them simply being "hard" samples, thus meaning unrelated models can make similar predictions to the watermark, leading to ambiguity problems.

struct the watermark set $\mathcal{W} \subset \mathcal{C}$ of size $k$ by solving $\mathcal{W} = \underset{\mathcal{S}\subset\mathcal{C},\,|\mathcal{S}|=k}{\arg\min} \sum_{\mathbf{x}_i\in\mathcal{S}} F_{\hat{y}_i}(\mathbf{x}_i)$. By embedding watermarks into these low-confidence samples, we aim to reduce potential accuracy degradation, as these samples already lie near class boundaries. This makes their decision assignments easier to shift and less likely to affect confident regions.

**Maximum Distance**. Candidates are chosen to maximize distance from all other watermarks. This criterion aims to minimize unintended interactions between watermarks by spatially isolating them. Embedding watermarks in distant and sparsely populated regions mitigates the risk of collateral decision boundary shifts affecting nearby candidates.

This corresponds to the maximum diversity problem (Lozano, Molina, and García-Martínez 2011), which is well-known to be NP-hard but also that an optimal answer to this problem is guaranteed to be less than twice as efficient as a greedy method that simply selects the furthest point from all the selected samples (Gonzalez 1985). To that end,

| Dataset | Avila | | | Img Seg. | | | Letter Recognition | | | optdigits | | | pendigits | | | Wine Quality | | | average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method \| Ratio | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 |
| Cluster (Conf) | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.880 | 0.993 | 0.500 | **1.000** | **1.000** | 0.250 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.792 | 0.980 | 0.999 |
| Cluster (Dist) | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.507 | 0.968 | 0.500 | **1.000** | **1.000** | 0.750 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.875 | 0.918 | 0.995 |
| Outlier (Conf) | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.875 | 0.720 | 0.996 | 0.500 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.896 | 0.953 | 0.999 |
| Outlier (Dist) | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.760 | **1.000** | **1.000** | **1.000** | **1.000** | 0.250 | **1.000** | **1.000** | **1.000** | 0.960 | **1.000** | 0.875 | 0.953 | **1.000** |
| Wrong (Conf) | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | **1.000** | **1.000** | — | **1.000** | **1.000** | — |
| Wrong (Dist) | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | **1.000** | **1.000** | — | **1.000** | **1.000** | — |
| Conf. (Conf) | **1.000** | 0.962 | **1.000** | **1.000** | **1.000** | **1.000** | 0.875 | **0.920** | 0.992 | 0.500 | 0.850 | **1.000** | 0.250 | 0.974 | **1.000** | **1.000** | **1.000** | **1.000** | 0.771 | 0.951 | 0.999 |
| Conf. (Dist) | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.625 | 0.907 | 0.996 | 0.500 | 0.850 | **1.000** | 0.750 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.812 | 0.959 | 0.999 |
| Random (Conf) | 0.667 | 0.830 | 0.975 | **1.000** | **1.000** | **1.000** | **1.000** | 0.480 | 0.917 | 0.000 | 0.750 | **1.000** | 0.500 | 0.895 | **1.000** | **1.000** | 0.960 | **1.000** | 0.694 | 0.819 | 0.982 |
| Random (Dist) | **1.000** | **1.000** | 0.996 | **1.000** | **1.000** | **1.000** | **1.000** | 0.507 | 0.953 | 0.000 | 0.800 | **1.000** | 0.250 | 0.974 | **1.000** | **1.000** | **1.000** | 0.996 | 0.708 | 0.880 | 0.991 |

Table 1: Watermarking effectiveness for $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$ and $d_{cand=train} = 5$. For brevity, method names are abbreviated as Cluster, Outlier, Wrong, Conf. (*Confidence Flip*), and Random (random selection baseline) with selection strategies noted in parentheses: (Conf) (lowest-confidence) and (Dist) (maximum-distance). This is repeated for all further results. Success rate is generally high for all methods, with the *Random Flip* baseline being lower than the other methods by a significant gap.

we apply this substantially faster and simpler solution, as this selection process serves primarily as a supplementary safeguard rather than a pivotal error reduction tool.

# 4 Experiments

We evaluate the effectiveness of our method through experiments conducted on a range of public datasets that are standard— Avila (Stefano et al. 2018), Image Segmentation (or Img Seg.) (UCI Machine Learning Repository 1990), Letter Recognition (Slate 1991), Wine Quality (Cortez et al. 2009), Optical Recognition of Handwritten Digits (or optdigits) (Xu, Krzyzak, and Suen 1992), and Pen-Based Recognition of Handwritten Digits (or pendigits) (Alpaydın and Alimoglu 1996). As detailed previously, there has been limited prior work specifically addressing watermarking for gradient boosting decision trees (GBDTs). Thus, we compare our approach against an intuitive baseline: randomly selected watermark candidates drawn from $\mathcal{D}_{\text{cand}}$. The watermarks $\mathcal{W}$ are chosen using the selection strategies described in Section 3.4.

We test our method under two distinct scenarios, also described in Section 3.1. In the first, watermark samples are drawn from the same dataset used for training, i.e., $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$. This setting reflects cases where watermarking is performed internally. In the second, watermarking is performed using a dataset disjoint from the training set. This simulates scenarios where watermarking is applied post hoc—e.g., for outsourced or externally maintained models. In this case, the full dataset is partitioned into four disjoint subsets: $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{cand}}$, $\mathcal{D}_{\text{test}}$, and $\mathcal{D}_{\text{fine}}$. The training set is split 80:20 to form $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{cand}}$. For both cases, the test set is split 80:20 to yield $\mathcal{D}_{\text{test}}$ and $\mathcal{D}_{\text{fine}}$, which is used in fine-tuning w.r.t. Section 4.3. If a dataset provides distinct training and test sets, we use them directly. Otherwise, we partition the dataset using an 80:20 train-test split. As detailed in Section 3.2, for $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$, $\mathcal{W}$ is duplicated. We set this duplication factor $d_{cand=train} = 5$.

We assume all watermark bits in $\mathcal{W}$ are set to 1, providing a stress test for effectiveness and robustness. We evaluate watermark ratios ($|\mathcal{W}|/|\mathcal{D}_{train}|$) of 0.001, 0.01, and 0.1 across scenarios, sampling strategies (Section 3.3), and selection strategies (Section 3.4). Unless otherwise specified, all models are trained for 200 iterations during all phases, and we set $|\mathcal{C}| = 2|\mathcal{W}|$. Below, we describe the characteristics under consideration, the experimental setup, and results.

## 4.1 Watermarking Effectiveness

As we do not manipulate the model itself directly to insert the watermark, we do not have an innate guarantee that the watermarks are correctly embedded post training. To show that our method can embed watermarks with reasonable reliability, we apply the following process: An initial model is trained on the $\mathcal{D}_{\text{train}}$ dataset, which in turn is used for prediction on the on the $\mathcal{D}_{\text{cand}}$ dataset to identify $n$ candidate samples adhering to the requirements detailed in Section 3.3. A subset of size $k$ of these candidates is selected and modified following the approaches detailed in Section 3.4. The initial model is then watermarked (i.e. fine-tuned) using these samples—i.e. with the dataset $\mathcal{W}' = \{c'_i = (\mathbf{x}_i, y_i^{\text{wm}}) \mid (\mathbf{x}_i, y_i) \in \mathcal{W},\ y_i^{\text{wm}} \neq y_i\}$, where $y_i^{\text{wm}}$ is as defined by Equation 2.

Watermarking effectiveness, denoted $\mathcal{A}_{\text{wm}}$, is measured by the proportion of watermark samples identified as the watermark label by the watermarked model $F^{\text{wm}}$:

$$\mathcal{A}_{\text{wm}} = \frac{1}{|\mathcal{W}'|} \sum_{(\mathbf{x}_i, y_i^{\text{wm}}) \in \mathcal{W}'} \mathbf{1}(F^{\text{wm}}(\mathbf{x}_i) = y_i^{\text{wm}}) \qquad (7)$$

This effectiveness can be increased arbitrarily by duplicating $\mathcal{W}'$ in the fine-tuning dataset during watermarking, but at the cost of other issues such as model accuracy. We do not apply this in all further testing, to prevent overfitting to watermark samples obscuring relative performance.

The results for $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$ are presented in Table 1,and those where $\mathcal{D}_{\text{cand}} \neq \mathcal{D}_{\text{train}}$ in Table 3. In both cases, general success rates are good; *Wrong Prediction Flip* also has very high success rates where it is applicable, but the aspect that it requires samples that yield wrong predictions significantly limits the number of watermarks that are applicable, especially for $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$, as outlined in Section 3.3.

## 4.2 General Accuracy

Maintaining the general predictive performance of the model after watermark embedding is critical, as the practical utility of a watermarking scheme depends heavily on minimizing accuracy degradation for non-watermarked data. We evaluate the general accuracy, $\mathcal{A}_{\text{model}}$, by assessing the performance

| Strategy | Recommended Context | Performance | Proposed Mechanism |
|---|---|---|---|
| **Cluster Center Flip** | $\mathcal{D}_{cand} \neq \mathcal{D}_{train}$ (i.e. the watermarking dataset is separate from the training dataset) | Shows high adjusted model accuracy (Table 5) and good effectiveness and confidence (Table 3, 7) w.r.t. $\mathcal{D}_{cand} \neq \mathcal{D}_{train}$ and Lowest Confidence candidate selection (see Section 3.4) | $\mathcal{D}_{\text{cand}} \neq \mathcal{D}_{\text{train}}$ (and Lowest Confidence selection as in Section 3.4) helps watermarks avoid entanglement between the candidate clusters and confident learned boundaries, which helps anchoring minimize the effect of the embedding on non-watermark samples, as well as make distancing clusters less important. |
| **Confidence Flip** | $\mathcal{D}_{cand} = \mathcal{D}_{train}$ (i.e. the watermarking dataset overlaps with the training dataset). | Shows good relative performance, especially for $\mathcal{D}_{cand} = \mathcal{D}_{train}$. | Targeting lower confidence samples eases the effort in "overwriting" the sample, and thus leads to better effectiveness and resilience. |
| **Outlier Flip** | Low Watermarking Rates (e.g., $\lvert\mathcal{W}\rvert/\lvert\mathcal{D}_{train}\rvert = 0.001$). | Generally effective at low rates, but less competitive at higher rates (e.g. $\lvert\mathcal{W}\rvert/\lvert\mathcal{D}_{train}\rvert = 0.1$). | Spatial isolation of "outlier" samples is lost at high watermarking rates, reducing effectiveness. |
| **Wrong Prediction Flip** | Datasets with enough wrongly predicted samples for a valid $\mathcal{D}_{cand}$. | Good performance w.r.t. most metrics when viable. | Only viable when a large supply of misclassified samples is abundant, which is heavily dataset-dependent - and generally only $\mathcal{D}_{cand} \neq \mathcal{D}_{train}$ (see validity between Table 3 and Table 1) due to tendency of GBDT to overfit to training dataset. |

Table 2: Summary of data contexts, empirical performance, and proposed mechanisms based on experimental results in Section 4.

of the fine-tuned model on the standard test set $\mathcal{D}_{\text{test}}$:

$$\mathcal{A}_{\text{model}} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{test}}} \mathbf{1}\left(F^{\text{wm}}(\mathbf{x}_i) = y_i\right) \quad (8)$$

We emphasize that $\mathcal{A}_{\text{model}}$ alone does not capture practical watermarking effectiveness. Intuitively, methods with lower $\mathcal{A}_{\text{wm}}$ would induce smaller perturbations to the original model and consequently preserve higher $\mathcal{A}_{\text{model}}$; however, this reduced deviation is at the expense of weaker watermark embedding, limiting utility in practice. Thus, we define the *adjusted model accuracy* as $\mathcal{A}'_{\text{model}} = \mathcal{A}_{\text{model}} \cdot \mathcal{A}_{\text{wm}}$, which jointly considers generalization and watermark strength to provide a more comprehensive evaluation. Table 4 presents results for the case where $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$, while Table 5 reports on the setting with $\mathcal{D}_{\text{cand}} \neq \mathcal{D}_{\text{train}}$.

For $\mathcal{D}_{\text{cand}} \neq \mathcal{D}_{\text{train}}$ we observe that *Cluster Flip* and *Wrong Prediction Flip* generally yield the highest adjusted model accuracy, provided the latter has access to a sufficient number of incorrect predictions. These methods are explicitly designed to minimize their impact on generalization: *Cluster Flip* anchors watermarks near existing decision boundaries using nearest-neighbor heuristics, while *Wrong Prediction Flip* restricts modifications to already misclassified samples. In contrast, *Outlier Flip* and *Confidence Flip* only apply the implicit expectation that that outliers or low-confidence samples will generally occupy sparsely populated regions of the feature space for this purpose. For $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$ this is less visible, although a gap between the methods and baseline is still visible especially for $|\mathcal{W}|/|\mathcal{D}_{train}| = 0.01$ and $0.1$.

### 4.3 Fine-tuning Robustness

A central concern of our work is watermark robustness to further fine-tuning. To evaluate this, we fine-tune the watermarked model $F^{\text{wm}}$ using the $\mathcal{D}_{\text{fine}}$ dataset. This simulates scenarios where the model is modified after deployment. We define robustness as the proportion of correctly embedded watermark samples that remain intact (i.e., still trigger the intended model response) after such tuning, i.e. for the further fine-tuned watermarked model $F^{\text{wm}'}$:

$$\text{Fine-tuning robustness} = \frac{\sum_{(\mathbf{x}_i, y_i^{\text{wm}}) \in \mathcal{W}'} \mathbf{1}\left(F^{\text{wm}}(\mathbf{x}_i) = y_i^{\text{wm}} \wedge F^{\text{wm}'}(\mathbf{x}_i) = y_i^{\text{wm}}\right)}{\sum_{(\mathbf{x}_i, y_i^{\text{wm}}) \in \mathcal{W}'} \mathbf{1}\left(F^{\text{wm}}(\mathbf{x}_i) = y_i^{\text{wm}}\right)} \quad (9)$$

As Eq. 9 considers only successful watermark predictions, we do not adjust for effectiveness again. Table 6 reports results where $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$, while Table 7 shows outcomes where $\mathcal{D}_{\text{cand}} \neq \mathcal{D}_{\text{train}}$. The proposed methods generally show good robustness versus the baseline.

### 4.4 Selection Strategy

Based on our results in Section 4, we synthesize our findings to strategic guidance in relation to the contexts and proposed mechanisms by which each embedding strategy performs best, as well as our recommendations in Table 2.

## 5 Conclusion

We propose the first robust watermarking framework for GBDT models, addressing their sequential and non-differentiable structure. Using in-place fine-tuning and four embedding strategies, we show that resilient watermarks can be embedded with minimal impact on performance. Our methods achieve high success rates, preserve accuracy, and remain robust to fine-tuning.


## Acknowledgments

This work is partially supported by the National Science Foundation award 2247619.

| Dataset | Avila | | | Img Seg. | | | Letter Recognition | | | optdigits | | | pendigits | | | Wine Quality | | | average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method \| Ratio | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 |
| Cluster (Conf) | **1.000** | 0.881 | 0.871 | **1.000** | **1.000** | **1.000** | **1.000** | 0.833 | 0.930 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.850 | **1.000** | **1.000** | 0.927 | **0.967** |
| Cluster (Dist) | **1.000** | **1.000** | 0.990 | **1.000** | **1.000** | **1.000** | 0.667 | 0.517 | 0.723 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.500 | 0.900 | 0.933 | 0.861 | 0.903 | 0.941 |
| Outlier (Conf) | **1.000** | **1.000** | 0.967 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.538 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.950 | 0.672 | **1.000** | 0.992 | 0.863 |
| Outlier (Dist) | **1.000** | 0.976 | **0.993** | **1.000** | **1.000** | **1.000** | **1.000** | 0.633 | 0.490 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.950 | 0.682 | **1.000** | 0.927 | 0.861 |
| Wrong (Conf) | — | — | — | **1.000** | **1.000** | — | **1.000** | **1.000** | — | **1.000** | — | — | **1.000** | — | — | **1.000** | **1.000** | — | **1.000** | **1.000** | — |
| Wrong (Dist) | — | — | — | **1.000** | **1.000** | — | **1.000** | **1.000** | — | **1.000** | — | — | **1.000** | — | — | **1.000** | 0.950 | — | **1.000** | 0.983 | — |
| Conf. (Conf) | **1.000** | **1.000** | 0.734 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **0.972** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.974 | **1.000** | **1.000** | 0.947 |
| Conf. (Dist) | **1.000** | **1.000** | 0.828 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.880 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.851 | **1.000** | **1.000** | 0.927 |
| Random (Conf) | **1.000** | 0.810 | 0.670 | **1.000** | **1.000** | **1.000** | **1.000** | 0.817 | 0.487 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.850 | 0.646 | **1.000** | 0.913 | 0.800 |
| Random (Dist) | **1.000** | 0.952 | 0.888 | **1.000** | **1.000** | **1.000** | **1.000** | 0.750 | 0.495 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.800 | 0.713 | **1.000** | 0.917 | 0.849 |

Table 3: Watermarking effectiveness for $\mathcal{D}_{\text{cand}} \neq \mathcal{D}_{\text{train}}$, without $\mathcal{W}'$ duplication. On average *Confidence Flip* has the highest success rates, then *Wrong Prediction Flip*, insofar as where it is valid.

| Dataset | Avila | | | Img Seg. | | | Letter Recognition | | | optdigits | | | pendigits | | | Wine Quality | | | average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method \| Ratio | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 |
| Cluster (Conf) | 0.998 | 0.996 | 0.939 | **0.846** | 0.846 | 0.827 | 0.967 | 0.848 | 0.928 | 0.486 | **0.977** | 0.960 | 0.243 | **0.973** | 0.961 | 0.655 | 0.641 | 0.618 | 0.699 | **0.880** | 0.872 |
| Cluster (Dist) | 0.998 | 0.998 | 0.965 | **0.846** | 0.846 | 0.827 | **0.968** | 0.489 | 0.900 | 0.487 | 0.974 | **0.965** | 0.729 | 0.970 | 0.961 | 0.655 | 0.647 | 0.609 | 0.781 | 0.820 | 0.871 |
| Outlier (Conf) | 0.998 | 0.996 | 0.950 | **0.846** | 0.846 | 0.827 | 0.849 | 0.693 | 0.917 | 0.488 | 0.971 | 0.945 | **0.973** | **0.973** | 0.940 | 0.659 | 0.645 | 0.632 | **0.802** | 0.854 | 0.869 |
| Outlier (Dist) | 0.998 | 0.996 | 0.957 | **0.846** | 0.846 | 0.827 | 0.963 | 0.732 | 0.933 | **0.974** | 0.970 | 0.947 | 0.243 | 0.972 | 0.964 | 0.664 | 0.615 | 0.637 | 0.781 | 0.855 | 0.877 |
| Wrong (Conf) | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0.656 | 0.639 | — | 0.656 | 0.639 | — |
| Wrong (Dist) | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0.655 | 0.650 | — | 0.655 | 0.650 | — |
| Conf. (Conf) | 0.999 | 0.957 | 0.945 | **0.846** | 0.846 | 0.827 | 0.846 | **0.885** | 0.936 | 0.487 | 0.827 | 0.963 | 0.243 | 0.945 | 0.966 | **0.667** | **0.660** | 0.641 | 0.681 | 0.854 | 0.880 |
| Conf. (Dist) | 0.998 | 0.995 | 0.949 | **0.846** | 0.846 | 0.846 | 0.606 | 0.876 | **0.937** | 0.487 | 0.828 | 0.963 | 0.730 | 0.971 | 0.965 | **0.667** | 0.653 | 0.638 | 0.722 | 0.862 | **0.883** |
| Random (Conf) | 0.666 | 0.829 | 0.950 | **0.846** | **0.865** | **0.865** | 0.965 | 0.462 | 0.869 | 0.000 | 0.732 | 0.964 | 0.485 | 0.866 | 0.965 | 0.658 | 0.620 | **0.649** | 0.603 | 0.729 | 0.877 |
| Random (Dist) | **0.999** | **0.998** | **0.975** | **0.846** | **0.865** | 0.846 | 0.962 | 0.487 | 0.897 | 0.000 | 0.778 | 0.963 | 0.243 | 0.944 | **0.968** | 0.651 | 0.642 | 0.636 | 0.617 | 0.786 | 0.881 |

Table 4: Adjusted model accuracy for $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$ and $d_{cand=train} = 5$. *Cluster Flip*, *Confidence Flip*, *Outlier Flip* all show competitive results.

| Dataset | Avila | | | Img Seg. | | | Letter Recognition | | | optdigits | | | pendigits | | | Wine Quality | | | average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method \| Ratio | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 |
| Cluster (Conf) | 0.998 | 0.878 | 0.848 | **0.846** | **0.846** | **0.846** | 0.961 | 0.803 | 0.878 | 0.973 | 0.972 | 0.958 | **0.973** | 0.971 | 0.963 | 0.629 | 0.544 | **0.614** | **0.897** | 0.836 | **0.851** |
| Cluster (Dist) | 0.997 | **0.996** | 0.965 | **0.846** | **0.846** | 0.827 | 0.642 | 0.498 | 0.682 | 0.972 | 0.974 | 0.957 | 0.971 | 0.971 | 0.959 | 0.318 | 0.557 | 0.573 | 0.791 | 0.807 | 0.827 |
| Outlier (Conf) | 0.997 | 0.994 | 0.944 | 0.827 | 0.827 | 0.827 | 0.956 | 0.937 | 0.510 | 0.970 | 0.965 | 0.949 | 0.972 | 0.970 | 0.961 | 0.627 | 0.591 | 0.422 | 0.891 | 0.881 | 0.769 |
| Outlier (Dist) | 0.997 | 0.971 | **0.971** | 0.827 | 0.827 | 0.788 | 0.955 | 0.601 | 0.466 | 0.972 | 0.965 | 0.952 | 0.970 | 0.969 | 0.961 | 0.629 | 0.587 | 0.422 | 0.892 | 0.820 | 0.760 |
| Wrong (Conf) | — | — | — | **0.846** | **0.846** | — | 0.965 | 0.957 | — | **0.975** | — | — | 0.971 | — | — | 0.632 | **0.637** | — | 0.878 | 0.813 | — |
| Wrong (Dist) | — | — | — | **0.846** | **0.846** | — | **0.966** | **0.961** | — | **0.975** | — | — | 0.969 | — | — | **0.639** | 0.597 | — | 0.879 | 0.801 | — |
| Conf. (Conf) | 0.997 | 0.992 | 0.708 | 0.827 | 0.827 | 0.827 | 0.963 | 0.957 | **0.915** | 0.971 | 0.972 | 0.967 | 0.971 | 0.969 | 0.965 | 0.631 | 0.623 | 0.598 | 0.893 | 0.890 | 0.830 |
| Conf. (Dist) | **0.998** | 0.995 | 0.807 | 0.827 | 0.827 | 0.827 | 0.964 | 0.956 | 0.831 | 0.969 | 0.969 | 0.968 | 0.971 | 0.970 | 0.961 | 0.631 | 0.634 | 0.520 | 0.893 | **0.892** | 0.819 |
| Random (Conf) | 0.997 | 0.807 | 0.660 | 0.827 | 0.827 | 0.827 | 0.958 | 0.778 | 0.465 | 0.974 | **0.975** | 0.965 | 0.969 | 0.973 | 0.959 | 0.633 | 0.533 | 0.408 | 0.893 | 0.815 | 0.714 |
| Random (Dist) | 0.996 | 0.949 | 0.873 | 0.827 | 0.827 | 0.827 | 0.956 | 0.710 | 0.471 | 0.974 | 0.973 | **0.969** | 0.970 | **0.973** | **0.965** | 0.637 | 0.502 | 0.438 | 0.893 | 0.822 | 0.757 |

Table 5: Adjusted model accuracy for selected watermarking rates, for $\mathcal{D}_{\text{cand}} \neq \mathcal{D}_{\text{train}}$, without $\mathcal{W}'$ duplication. *Cluster Center Flip* has a relatively high accuracy.

| Dataset | Avila | | | Img Seg. | | | Letter Recognition | | | optdigits | | | pendigits | | | Wine Quality | | | average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method \| Ratio | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 |
| Cluster (Conf) | **1.000** | **1.000** | 0.989 | **1.000** | **1.000** | **1.000** | 0.750 | 0.747 | 0.781 | 0.500 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.875 | 0.958 | 0.962 |
| Cluster (Dist) | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.875 | 0.400 | 0.585 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.980 | 0.979 | 0.900 | 0.927 |
| Outlier (Conf) | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.625 | 0.560 | 0.755 | 0.000 | 0.950 | **1.000** | 0.250 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.646 | 0.918 | 0.959 |
| Outlier (Dist) | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.547 | 0.540 | 0.000 | 0.800 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.833 | 0.891 | 0.923 |
| Wrong (Conf) | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | **1.000** | **1.000** | — | **1.000** | **1.000** | - |
| Wrong (Dist) | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | **1.000** | **1.000** | — | **1.000** | **1.000** | - |
| Conf. (Conf) | **1.000** | **1.000** | 0.998 | **1.000** | **1.000** | **1.000** | **1.000** | **0.960** | 0.919 | 0.500 | 0.900 | **1.000** | 0.500 | 0.947 | **1.000** | **1.000** | **1.000** | **1.000** | 0.833 | 0.968 | 0.986 |
| Conf. (Dist) | **1.000** | **1.000** | 0.990 | **1.000** | **1.000** | **1.000** | **1.000** | 0.907 | **0.947** | 0.500 | 0.950 | 0.995 | 0.500 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.833 | 0.976 | **0.989** |
| Random (Conf) | 0.667 | 0.962 | 0.939 | **1.000** | **1.000** | **1.000** | 0.750 | 0.467 | 0.605 | 0.500 | 0.800 | **1.000** | 0.750 | **1.000** | **1.000** | **1.000** | 0.960 | 0.992 | 0.778 | 0.865 | 0.923 |
| Random (Dist) | **1.000** | 0.981 | 0.989 | **1.000** | **1.000** | **1.000** | 0.750 | 0.400 | 0.616 | 0.000 | 0.850 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.792 | 0.872 | 0.934 |

Table 6: Robustness for $\mathcal{D}_{\text{cand}} = \mathcal{D}_{\text{train}}$ and $d_{cand=train} = 5$. All proposed methods show good performance, although the results at $|\mathcal{W}|/|\mathcal{D}_{train}| = 0.01$ show higher variance, likely due to the small number of watermarks involved. *Confidence Flip* is overall slightly more resilent than other methods, although the difference is limited.

| Dataset | Avila | | | Img Seg. | | | Letter Recognition | | | optdigits | | | pendigits | | | Wine Quality | | | average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method \| Ratio | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 | 0.001 | 0.01 | 0.1 |
| Cluster (Conf) | **1.000** | 0.860 | 0.717 | **1.000** | **1.000** | **1.000** | **1.000** | 0.556 | 0.811 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.765 | **0.979** | **1.000** | 0.863 | **0.918** |
| Cluster (Dist) | **1.000** | **1.000** | **0.983** | **1.000** | **1.000** | **1.000** | 0.444 | 0.233 | 0.450 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.250 | 0.855 | 0.871 | 0.782 | 0.848 | 0.884 |
| Outlier (Conf) | **1.000** | 0.952 | 0.913 | **1.000** | **1.000** | **1.000** | 0.333 | 0.433 | 0.240 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.760 | 0.462 | 0.889 | 0.858 | 0.769 |
| Outlier (Dist) | **1.000** | 0.976 | 0.967 | **1.000** | **1.000** | **1.000** | 0.667 | 0.348 | 0.194 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.760 | 0.462 | 0.944 | 0.847 | 0.770 |
| Wrong (Conf) | — | — | — | **1.000** | **1.000** | — | **1.000** | 0.967 | — | **1.000** | — | — | **1.000** | — | — | **1.000** | 0.950 | — | **1.000** | 0.972 | — |
| Wrong (Dist) | — | — | — | **1.000** | **1.000** | — | **1.000** | 0.950 | — | **1.000** | — | — | **1.000** | — | — | **1.000** | 0.902 | — | **1.000** | 0.951 | — |
| Conf. (Conf) | **1.000** | 0.833 | 0.525 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **0.881** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.919 | **1.000** | 0.972 | 0.888 |
| Conf. (Dist) | **1.000** | **1.000** | 0.620 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.697 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.712 | **1.000** | **1.000** | 0.838 |
| Random (Conf) | 0.800 | 0.636 | 0.442 | **1.000** | **1.000** | **1.000** | 0.333 | 0.381 | 0.212 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.500 | 0.680 | 0.447 | 0.772 | 0.783 | 0.684 |
| Random (Dist) | **1.000** | 0.952 | 0.747 | **1.000** | **1.000** | **1.000** | 0.667 | 0.375 | 0.214 | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | **1.000** | 0.560 | 0.483 | 0.944 | 0.815 | 0.741 |

Table 7: Robustness for $\mathcal{D}_{\text{cand}} \neq \mathcal{D}_{\text{train}}$. Similarly to Table 6, the results at $|\mathcal{W}|/|\mathcal{D}_{train}| = 0.01$ show high variance.

## References


Adi, Y.; Baum, C.; Cissé, M.; Pinkas, B.; and Keshet, J. 2018. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In Enck, W.; and Felt, A. P., eds., *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, 1615–1631. USENIX Association.

Alpaydın, E.; and Alimoglu, F. 1996. Pen-Based Recognition of Handwritten Digits. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5MG6K.

Calzavara, S.; Cazzaro, L.; Gera, D.; and Orlando, S. 2025. Watermarking Decision Tree Ensembles. In Simitsis, A.; Kemme, B.; Queralt, A.; Romero, O.; and Jovanovic, P., eds., *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025*, 569–575. OpenProceedings.org.

Chen, H.; Zhang, H.; Si, S.; Li, Y.; Boning, D. S.; and Hsieh, C. 2019. Robustness Verification of Tree-based Models. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 12317–12328.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In Krishnapuram, B.; Shah, M.; Smola, A. J.; Aggarwal, C. C.; Shen, D.; and Rastogi, R., eds., *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 785–794. ACM.

Cinà, A. E.; Grosse, K.; Demontis, A.; Vascon, S.; Zellinger, W.; Moser, B. A.; Oprea, A.; Biggio, B.; Pelillo, M.; and Roli, F. 2023. Wild Patterns Reloaded: A Survey of Machine Learning Security against Training Data Poisoning. *ACM Comput. Surv.*, 55(13s): 294:1–294:39.

Cortez, P.; Cerdeira, A.; Almeida, F.; Matos, T.; and Reis, J. 2009. Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst.*, 47(4): 547–553.

Ester, M.; Kriegel, H.; Sander, J.; and Xu, X. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In Simoudis, E.; Han, J.; and Fayyad, U. M., eds., *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, 226–231. AAAI Press.

Fan, T.; Chen, W.; Ma, G.; Kang, Y.; Fan, L.; and Yang, Q. 2024. SecureBoost+: Large Scale and High-Performance Vertical Federated Gradient Boosting Decision Tree. In Yang, D.; Xie, X.; Tseng, V. S.; Pei, J.; Huang, J.; and Lin, J. C., eds., *Advances in Knowledge Discovery and Data Mining - 28th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2024, Taipei, Taiwan, May 7-10, 2024, Proceedings, Part III*, volume 14647 of *Lecture Notes in Computer Science*, 237–249. Springer.

Fuhrer, B.; Tessler, C.; and Dalal, G. 2024. Gradient Boosting Reinforcement Learning. *CoRR*, abs/2407.08250.

Gonzalez, T. F. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theor. Comput. Sci.*, 38: 293–306.

Guo, J.; Li, Y.; Wang, L.; Xia, S.; Huang, H.; Liu, C.; and Li, B. 2023. Domain Watermark: Effective and Harmless Dataset Copyright Protection is Closed at Hand. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Iosipoi, L.; and Vakhrushev, A. 2022. SketchBoost: Fast Gradient Boosted Decision Tree for Multioutput Problems. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 3146–3154.

Law, A.; Leung, C.; Poddar, R.; Popa, R. A.; Shi, C.; Sima, O.; Yu, C.; Zhang, X.; and Zheng, W. 2020. Secure Collaborative Training and Inference for XGBoost. In Zhang, B.; Popa, R. A.; Zaharia, M.; Gu, G.; and Ji, S., eds., *PPMLP'20: Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice, Virtual Event, USA, November, 2020*, 21–26. ACM.

Lozano, M.; Molina, D.; and García-Martínez, C. 2011. Iterated greedy for the maximum diversity problem. *Eur. J. Oper. Res.*, 214(1): 31–38.

Lu, W.; Huang, Z.; Zhang, Q.; Wang, Y.; and Hong, C. 2023. Squirrel: A Scalable Secure Two-Party Computation Framework for Training Gradient Boosting Decision Tree. In Calandrino, J. A.; and Troncoso, C., eds., *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, 6435–6451. USENIX Association.

McElfresh, D. C.; Khandagale, S.; Valverde, J.; C., V. P.; Ramakrishnan, G.; Goldblum, M.; and White, C. 2023. When Do Neural Nets Outperform Boosted Trees on Tabular Data? In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Pagnotta, G.; Hitaj, D.; Hitaj, B.; Pérez-Cruz, F.; and Mancini, L. V. 2024. TATTOOED: A Robust Deep Neural Network Watermarking Scheme based on Spread-Spectrum Channel Coding. In *Annual Computer Security Applications Conference, ACSAC 2024, Honolulu, HI, USA, December 9-13, 2024*, 1245–1258. IEEE.

Rigaki, M.; and García, S. 2024. A Survey of Privacy Attacks

in Machine Learning. *ACM Comput. Surv.*, 56(4): 101:1–101:34.

Rouhani, B. D.; Chen, H.; and Koushanfar, F. 2018. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. *IACR Cryptol. ePrint Arch.*, 311.

Slate, D. 1991. Letter Recognition. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5ZP40.

Stefano, C.; Fontanella, F.; Maniaci, M.; and Freca, A. 2018. Avila. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5K02X.

Taha, K. 2025. Machine learning in biomedical and health big data: a comprehensive survey with empirical and experimental insights. *J. Big Data*, 12(1).

Uchida, Y.; Nagai, Y.; Sakazawa, S.; and Satoh, S. 2017. Embedding Watermarks into Deep Neural Networks. In Ionescu, B.; Sebe, N.; Feng, J.; Larson, M. A.; Lienhart, R.; and Snoek, C., eds., *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR 2017, Bucharest, Romania, June 6-9, 2017*, 269–277. ACM.

UCI Machine Learning Repository. 1990. Image Segmentation. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5GP4N.

Xu, L.; Krzyzak, A.; and Suen, C. Y. 1992. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. Syst. Man Cybern.*, 22(3): 418–435.

Yan, Y.; Pan, X.; Zhang, M.; and Yang, M. 2023. Rethinking White-Box Watermarks on Deep Learning Models under Neural Structural Obfuscation. In Calandrino, J. A.; and Troncoso, C., eds., *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, 2347–2364. USENIX Association.

Zhao, W.; Lao, Y.; and Li, P. 2022. Integrity Authentication in Tree Models. In Zhang, A.; and Rangwala, H., eds., *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, 2585–2593. ACM.