TTTS: Tree Test Time Simulation for Enhancing Decision Tree Robustness against Adversarial Examples

Seffi Cohen, Ofir Arbili, Yisroel Mirsky, Lior Rokach

Ben-Gurion University of the Negev
seffi@post.bgu.ac.il, arbili@post.bgu.ac.il

Abstract

Decision trees are widely used for addressing learning tasks involving tabular data. Yet, they are susceptible to adversarial attacks. In this paper, we present Tree Test Time Simulation (TTTS), a novel inference-time methodology that incorporates Monte Carlo simulations into decision trees to enhance their robustness. TTTS introduces a probabilistic modification to the decision path, without altering the underlying tree structure. Our comprehensive empirical analysis of 50 datasets yields promising results. Without the presence of any attacks, TTTS has successfully improved model performance from an AUC of 0.714 to 0.773. Under the challenging conditions of white-box attacks, TTTS demonstrated its robustness by boosting performance from an AUC of 0.337 to 0.680. Even when subjected to black-box attacks, TTTS maintains high accuracy and enhances the model's performance from an AUC of 0.628 to 0.719. Compared to defenses such as Feature Squeezing, TTTS proves to be much more effective. We also found that TTTS exhibits similar robustness in decision forest settings across different attacks.

Introduction

Decision Tree (DT) models play a crucial role in various fields due to their transparency, interpretability, and ability to handle complex decision-making processes (Costa and Pedreira 2022). They provide intuitive insights into how a model arrives at its conclusions, making them valuable for both analysis and application. However, DT models are susceptible to adversarial examples (Vos and Verwer 2021; Chen et al. 2019), where small, carefully crafted perturbations can lead to significant misclassifications. Moreover, DT models can be exploited in both white-box and black-box settings where the attacker either has full knowledge or no knowledge of the models parameters (Chen et al. 2019). These attacks compromise the performance and reliability of the models, posing substantial risks (Karchmer 2023), underscoring the need for enhanced robustness techniques to ensure their reliable performance in real-world scenarios.

Although many works have suggested ways for making DT models robust against adversarial attacks (Vos and Verwer 2022b; Ranzato and Zanella 2021; Calzavara et al. 2020; Yang et al. 2020; Guo et al. 2022; Vos and Verwer 2022a) all of these solutions modify the model or the training process. This is a disadvantage for several reasons: (1) changing model parameters or the values of the node's threshold in case of DT, often harm a model's performance on clean data (Andriushchenko and Hein 2019; Chen et al. 2019; Vos and Verwer 2021), (2) these defences cannot be applied to deployed models, and (3) some of these methods cannot be applied to every kind of DT which limits a developer's options. This holds particular significance for mission-critical systems. For example, during the Covid-19 pandemic, a DT-based model called pangoLEARN (Turakhia et al. 2021), was used by healthcare professionals and policymakers for timely and accurate lineage assignment of SARS-CoV-2 genomes. A decision tree was chosen due to its competitive predictive performance and hierarchical structure, closely resembling a phylogenetic tree. Nonetheless, employing a DT-based model introduces a plausible cyber-bioterrorism vulnerability (Muthuppalaniappan and Stevenson 2021). In this scenario, we would like to make the model robust to potential cyber-bioterrorism attacks while preserving its structure and parameters. The latter ensures the model maintains its competitive predictive performance and explainability. Existing methods that revise the model can lead to inaccurate lineage assignment of SARS-CoV-2 genomes, which may have severe consequences for healthcare professionals and policymakers.

In this work, we introduce a novel method, termed Tree Test Time Simulation (TTTS), targeted at enhancing the robustness of trained models without necessitating any alterations to the input data or the values of the node's threshold in the tree. TTTS exhibits the capacity to augment both the robustness and performance of DT on adversarial examples as well as on clean data, all the while preserving the model's inherent structure and training.

TTTS fundamentally integrates Monte Carlo Simulations into DT to effectively enhance test data, thereby improving the accuracy of predictions. For each test instance, the algorithm predominantly adheres to the correct branch, but intermittently deviates to an alternate branch based on a probabilistic measure. This probabilistic measure is determined by one of five unique strategies we propose, each catering to specific challenges intrinsic to DT predictions. By embedding simple statistical measures, TTTS ensures a minimal degree of plausibility between the decision at each level of the tree and the final classification outcome. A notable merit of this approach is its non-interference with the generation of the DT, ensuring compatibility with DT generated by a variety of algorithms, not limited to those from Scikit (Pedregosa et al. 2011).

Our method, integrated with the Adversarial Robustness Toolbox, was subjected to rigorous testing against white-box (DecisionTreeAttack) (Papernot et al. 2016) and black-box (ZooAttack) (Chen et al. 2017) attacks across a diverse collection of 50 datasets. The encouraging results highlight the potential of TTTS as a comprehensive defense mechanism, reinforcing the robustness of DT against adversarial attacks while enhancing prediction accuracy. Hence, our contribution is a significant step forward in safeguarding DT models, particularly in the context of tabular data, against adversarial threats.

Related Work

Various methods have been proposed to improve the robustness of machine learning models against adversarial examples, noise, and other perturbations. In this section, we discuss some of these methods and their characteristics.

1. **Provably Robust Boosted Decision Stumps and Trees against Adversarial Attacks** This method focuses on training provably robust boosted decision stumps and trees against adversarial attacks (Andriushchenko and Hein 2019). The approach involves formulating the problem as a convex optimization problem and solving it using a boosting algorithm. This method has been shown to provide robustness guarantees for decision stumps and trees.
2. **Efficient Training of Robust Decision Trees Against Adversarial Examples** This method aims to train robust DTs against adversarial examples efficiently (Vos and Verwer 2021). The approach involves using a novel splitting criterion that considers the worst-case adversarial perturbation of the input data. This method has been shown to improve the robustness of DTs against adversarial examples.
3. **Robust Optimal Classification Trees Against Adversarial Examples** This approach proposes robust optimal classification trees against adversarial examples (Vos and Verwer 2022b). The method involves formulating the problem as a mixed-integer linear program and solving it using a branch-and-bound algorithm. This approach has been shown to improve the robustness of classification trees against adversarial examples.
4. **Adversarially Robust Decision Tree Relabeling** This method focuses on adversarially robust DT relabeling (Vos and Verwer 2022a). The approach involves relabeling the leaves of a DT to minimize the worst-case adversarial loss. This method has been shown to improve the robustness of DTs against adversarial examples.
5. **Genetic Adversarial Training of Decision Trees** This approach proposes genetic adversarial training of decision trees (Ranzato and Zanella 2021). The method involves using a genetic algorithm to generate adversarial examples and training the DT on the augmented dataset. This approach has been shown to improve the robustness of DTs against adversarial attacks.
6. **TREANT: Training Evasion-Aware Decision Trees** TREANT is a method for training evasion-aware DTs (Calzavara et al. 2020). The approach involves incorporating an evasion-aware splitting criterion into the DT training process. This method has been shown to improve the robustness of DTs against evasion attacks.
7. **Fast Provably Robust Decision Trees and Boosting** This method focuses on fast provably robust DTs and boosting (Guo et al. 2022). The approach involves using a novel algorithm for training DTs that provides robustness guarantees against adversarial attacks. This method has been shown to improve the robustness of DTs and boosting algorithms against adversarial examples.
8. **Hardening Hardware Accelerartor Based CNN Inference Phase Against Adversarial Noises** Layer-wise prediction inconsistency involves obtaining model predictions at different layers of a CNN and detecting adversarial inputs based on prediction inconsistency across these layers (Odetola, Adeyemo, and Hasan 2022). This method can help improve the robustness of the model during the inference phase.
9. **Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks** Feature Squeezing (Xu, Evans, and Qi 2017) is a method for detecting adversarial examples in deep neural networks, particularly in the context of image data, but support also tabular data and DTs. The technique works by applying transformations to the input data, such as reducing the color bit depth of each pixel or applying spatial smoothing. The differences in the model's outputs between the original and transformed inputs are then used for detecting adversarial examples.

Methodology

TTTS integrates the concept of Monte Carlo Simulation into the DT architecture. For each test instance, TTTS mainly follows the correct branch during traversal but may sporadically deviate to an alternative branch, based on a predefined probability.The underlying intuition of TTTS resides in the chance to explore multiple paths within the tree, an opportunity that becomes amplified in scenarios involving heuristics, such as uncertainty.

Let T denote a DT model, I a test instance for prediction, $P_T(I)$ the prediction function of T for I, and $TTTS_{P,S}(T)$ the TTTS-enhanced prediction function of T with a probability method P and S simulations. Let $\Delta_{adv}(I)$ represent the adversarial perturbation function that generates adversarial instances, and $\mathcal{R}(P_T(I), \Delta_{adv}(I))$ be the robustness measure of the prediction function $P_T(I)$ against adversarial perturbation $\Delta_{adv}(I)$, where a higher value indicates higher robustness. In this context, for any DT model T, any test instance I, any of the five proposed probability methods P, and any number of simulations S, the application of Tree Test Time Simulation (TTTS) during the inference phase

Method	Tabular support	Only on test time	Improving on clean data	DT
1	✓	×	×	✓
2	✓	×	×	✓
3	✓	×	×	✓
4	✓	×	×	✓
5	✓	×	×	✓
6	✓	×	×	✓
7	✓	×	×	✓
8	×	✓	✓	×
9	✓	✓	×	✓
10 (Our)	✓	✓	✓	✓

Table 1: Comparison of TTTS with existing methods aimed at enhancing the robustness of machine learning models against adversarial example classification. The methods are evaluated based on their compatibility with tabular data, ability to implement without modifying the model (only on test time), the potential to improve performance on clean data (not just with adversarial examples), and support for DTs. Among all the examined techniques, only TTTS meets all the specified requirements.

enhances the robustness of the model's predictions against adversarial attacks as:

$$\mathcal{R}(TTTS_{P,S}(T), \Delta_{adv}(I)) > \mathcal{R}(P_T(I), \Delta_{adv}(I)) \quad (1)$$

Moreover, it's important to note that $TTTS_{P,S}(T)$ does not modify the original structure of T and improves performance against both black-box and white-box attacks. This study proposes five methods for DT node splitting, each applying a unique probability mechanism to determine the selection of split point candidates. The methods are as follows:

1. **Fixed Probability Method:** This method serves as a foundational benchmark for other techniques. At any tree node, we either follow the original tree's direction or opt for the opposite. This choice is random, with a preset fixed probability for selecting the inverse direction. As the tree is traversed, there's a significant likelihood of selecting the correct branch and a lower probability for choosing the incorrect one. This embodies the "Monte Carlo Simulation" principle in DT analysis.
   In this approach, the probability of a flip remains constant, determined beforehand. The formulation is as follows: Let p_{flip} denote the flip probability, and p_{fixed} be a predetermined constant. Then, the formula becomes:

$$p_{\text{flip}} = p_{\text{fixed}} \quad (2)$$

2. **Depth Probability Method:** The depth of a DT, indicative of the longest path from root to leaf, directly influences its performance and clarity. Its impact is threefold: *Complexity:* Deeper trees inherently increase in complexity due to more levels and nodes, complicating their interpretability, especially for very deep trees. *Accuracy:* Enhanced depth can better the model's accuracy on training data, allowing decisions at each node to exploit diverse data features, thereby unveiling intricate patterns. *Overfitting:* Greater depth can lead to overfitting, where a model overly tunes to training data noise, reducing its generalization. Deeper nodes, representing more specific rules, are more susceptible to this noise. As we venture further into DT algorithms, we often witness eroding confidence in thresholds due to decreasing sample sizes, overfitting, escalating complexity, and heightened variance: *Decreased Sample Size:* Successive splits in a tree yield ever-smaller data subsets, meaning deeper decisions rely on fewer examples, potentially destabilizing threshold reliability. *Overfitting:* Delving deeper can make the algorithm overly tailored to training data, risking overfitting. Such trees might learn noise, which can undermine confidence in deeper-level decision thresholds. *Increased Complexity:* A tree's complexity burgeons with depth, producing thresholds that may be less interpretable and reliable, especially if they hinge on intricate feature interplays. *Variance:* Deeper trees might encapsulate substantial data variance, possibly diminishing threshold confidence due to sensitivity to minor data perturbations. In essence, while deeper trees can discern intricate data patterns, they might simultaneously erode confidence in decision thresholds. Consequently, in our approach, we modulate the probability of branch selection based on node depth: the deeper the node, the more likely we opt for an alternative path.
   Here, the flip probability is proportional to the depth of the current node. The mathematical formulation is as follows: Let d denote the depth of the current node, $p_{\text{max\_depth}}$ be the maximum possible flip probability, and r_{depth} the depth rate. We calculate the flip probability as:

$$p_{\text{flip}} = \min(r_{\text{depth}} \cdot d, p_{\text{max\_depth}}) \quad (3)$$

3. **Agreement Probability Method:** DTs use nodes to depict decision junctures. The class distribution within these nodes profoundly influences the predictive prowess of the model. A node heavily skewed towards one class indicates the model's heightened assurance in classifying new entries into that dominant class. This assurance arises from the model's intrinsic "majority voting" logic, which leans towards the predominant class. The clearer this majority, the higher the prediction accuracy, especially for data akin to the training set. On the contrary, a node with a balanced class distribution signals ambivalence. In such scenarios, the model's confidence dips, occasionally resorting to pure conjecture. The very essence of DT algorithms is to minimize this uncertainty by preferring tests that result in uneven class distributions. Nodes with balanced classes can be easily misled compared to their majority-dominated counterparts. This approach computes probability by analyzing the label distribution in a node, emphasizing the ratio of the dominant label to the total samples in the node. Fundamentally, a dwindling dominance in label distribution amplifies the chances of deviating from the primary tree path. The intricate relationship between node class distribution and prediction precision remains pivotal in DT constructs. In this method, the flip probability is determined based on the proportion of the majority class. Let c_{max} be the count of the majority class at the node, n the

total number of samples at the node, and $p_{\text{max\_agreement}}$ the maximum possible flip probability. The flip probability is given by:

$$p_{\text{flip}} = \min\left(1 - \frac{c_{\max}}{n}, p_{\text{max\_agreement}}\right) \tag{4}$$

4. **Confidence Probability Method:** In a DT, a feature is chosen to split a node based on the range of the feature values present in that specific node. The variability in these values plays a crucial role in the tree's decision-making process. For instance, suppose we have a node where the splitting feature selected is "Age", aiming to predict whether someone will purchase a car. In this case, "Age" is chosen due to its broad array and variability in the data present at this node. Essentially, "Age" has a wide spread of values that supply sufficient information for an efficient split. The tree simply assesses a feature's ability to divide the data relying on its distribution. However, during the inference stage, we disregard the difference between the current split feature value and the average and standard deviation of the node samples. We presume that a larger gap between these two implies reduced confidence in the direction of the forthcoming branch. Based on this presumption, we propose the following method. We ascertain the probability of flipping the direction of the next branch by using the standard deviation of the splitting feature and the gap between the current value of this feature during the inference stage and the average for this specific feature in the node samples. This technique amplifies the probability of opting for the opposite direction when the discrepancy between the current value of the feature and its mean deviates significantly from its mean, gauged by the standard deviation of the node samples. In this approach, the flip probability is a function of the feature value's distance from the mean normalized by the standard deviation. Let v denote the feature value, μ the mean of feature values at the node, σ the standard deviation of feature values at the node, and $p_{\text{max\_confidence}}$ the maximum possible flip probability. The flip probability is given by:

$$p_{\text{flip}} = \max\left(p_{\text{max\_confidence}} - \frac{|v-\mu|}{\sigma+\epsilon}, 0\right) \tag{5}$$

Here, ϵ is a small constant added to prevent division by zero.

5. **Threshold Distance Probability Method:** Continuing from our previous method, we now factor in the threshold of the splitting feature in the current node. This is a shift in perspective where we consider not only the feature itself but also its relative position to the threshold in the current node. We determine the distance between the current feature value and the threshold, considering the standard deviation of the samples in the current node. Based on the distance computed, we then make a choice: either to flip the direction of the subsequent branch or to continue traversing as per the original tree. The smaller the distance, the higher the chance we will opt for the opposite direction in the next branch, as opposed to the path suggested by the original DT model. Essentially, this strategy presumes that a closer current feature value to the threshold leads to less certainty in deciding the appropriate direction for the next node or leaf.
In this method, the flip probability is based on the feature value's distance from the threshold, normalized by the standard deviation. Let t denote the threshold value, σ the standard deviation of feature values at the node, and $p_{\text{max\_threshold}}$ the maximum possible flip probability. The flip probability is given by:

$$p_{\text{flip}} = \max\left(p_{\text{max\_threshold}} - \frac{|v-t|}{\sigma+\epsilon}, 0\right) \tag{6}$$

Algorithm 1: $TTTS_{P,S}(T, I)$

Parameters: probability method P, S simulations
Input: test instance I, DT model T
Output: final_prediction

```
 1: predictions = []
 2: for $i = 1$ to $S$ do
 3:     $p_{\text{flip}} = P$
 4:     for each node in T.path do
 5:         if $(I.v < T.t) \neq (random[0,1] < p_{\text{flip}})$ then
 6:             chosen_branch = left branch
 7:         else
 8:             chosen_branch = right branch
 9:         end if
10:     end for
11:     predictions += traversed path prediction
12: end for
13: final_prediction = average(predictions)
14: return final_prediction
```

The TTTS algorithm (Algorithm 1) accepts the parameters P as the probability method which determines the likelihood of altering the decision path at each node, S as the number of simulations to be run, and the inputs: a I representing the data point to predict, and T as the DT model. The algorithm begins by initializing an empty list named $predictions$ (line 1) to store the outcomes of each simulation. For each simulation (line 2), the algorithm calculates a probability p_{flip} using the provided P (line 3). This p_{flip} represents the chance of diverting from the standard decision path at any given node in the T. The algorithm then enters a loop over each node in the tree's path for the given test instance (line 4). At each node, the decision whether to follow the standard decision path or deviate from it is evaluated (line 5). If the feature value of the test instance, $I.v$, is less than $T.t$, the feature threshold at the node and a random number between 0 and 1 is less than p_{flip}, or vice versa, the algorithm chooses the left branch (line 6). Otherwise, it chooses the right branch (line 8). After the tree traversal for each simulation, the prediction is made based on the leaf node reached and appended to the predictions list (line 11). Upon completion of all simulations, the algorithm averages the predictions in the predictions list to produce a $final\_prediction$ (line 13). This average ensures a consensus prediction from all individual simulation paths, thereby enhancing the robustness of the prediction. The $final\_prediction$ is then returned as the output of the algorithm (line 14).

The TTTS introduces a powerful technique for handling uncertainties, capitalizing on the strengths of Monte Carlo Simulation and probabilistic decision-making in DTs. These strategies provide promising avenues for handling adversarial attacks, significantly contributing to the robustness of decision tree-based models. In the following subsection, we will delve into further detail regarding the various strategies utilized for determining the probability of altering the path at each node, a critical component of the TTTS algorithm. Each of these methods presents a unique way of exploring the DT node splitting process, contributing to our understanding of model uncertainty in DT learning algorithms. Future sections will delve into their empirical validation.

Experiments

This section delineates the experimental settings, encompassing the datasets utilized, the experimental setup, and the evaluation metric. We performed extensive experiments to assess the robustness of TTTS against two types of model attacks – a white-box attack and a black-box attack. In addition, we contrasted our method with the Feature Squeezing defense strategy. Furthermore, we evaluated TTTS's efficacy on Random Forest (Breiman 2001) models, thereby investigating its capabilities on more robust models than DTs. The experiments were performed on 50 tabular classification datasets, applying TTTS with the probability heuristics suggested in the method section. These heuristics are as follows:

- Fixed: Fixed Probability Approach.
- Depth: Depth-Based Approach.
- Agree: Agreement-Based Approach.
- Conf: Confidence-Based Approach.
- Dist: Threshold Probability Approach.

The examined models were denoted as:

- DT: a standard DT model.
- RF: a Random Forest model.

In the context attacks, we considered:

- ZOO: a Zeroth Order Optimization black-box attack.
- DTA: a DT white-box attack.

As for the baseline, we included:

- FS: Feature Squeezing.

The scenario without our method was denoted as:

- w/o: without our method.

Datasets

Our experiments encompass a collection of 50 diverse datasets, each reflecting different real-world scenarios. The datasets contain a wide range of features, with a binary target variable always situated as the last column. The diversity in these datasets provides a robust environment for testing our model's scalability and versatility. To offer a visual overview of the datasets, we have plotted a scatter plot (Figure 1) where each point represents a dataset. The position

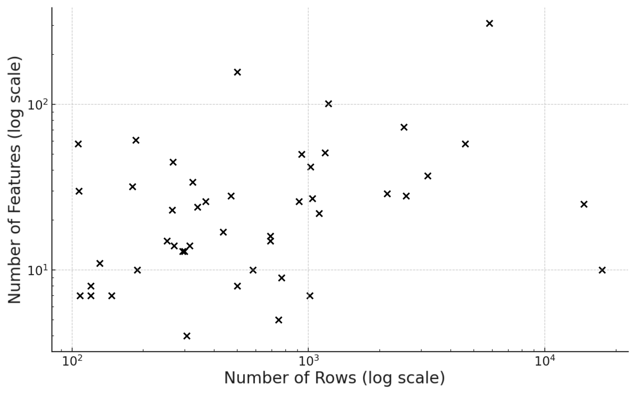

Figure 1: Scatter plot showing the number of rows VS the number of features for each evaluated dataset. Each point represents a dataset.

of the point on the x-axis indicates the number of instances (rows), while the position on the y-axis indicates the number of features (columns). Both axes are in logarithmic scale to balance the view of datasets with different scales. Figure 1 reveals that our datasets vary significantly in both scale (from 107 to 17442 instances) and complexity (from 7 to 309 features). This diversity enables a comprehensive evaluation of our proposed method's performance across different domains. Each dataset was subjected to necessary preprocessing steps like handling missing values, normalization, and categorical encoding as per its specific requirements. Further details about the exact preprocessing steps undertaken for each dataset can be found in the supplementary materials. Our objective across these diverse datasets is to showcase the robustness and general applicability of our proposed machine-learning model, regardless of the data's domain or scale.

Experimental Setup

In this study, we primarily focused on the application of our method to DT models but also expanded our exploration to random forest models. This extension allowed us to assess the enhancement of robustness in a model that is inherently more robust than DTs. We then compared the robustness of these models, under white-box and black-box attacks that specifically target tabular data, with that of an adversarial detection method. Below, we detail the experimental setup, including the configuration of the training and hyperparameters, model specifications, adversarial attacks, and the parameters for probability heuristics.

Classifiers We implemented our models using the DTs and Random Forests algorithms available in Scikit-learn (Pedregosa et al. 2011). For both algorithms, we trained the models using the Scikit default parameters with two exceptions: the random_state parameter, which was modified to ensure result reproducibility, and the n_estimators parameter, which was capped at 10 due to computational time constraints.

Probability Heuristics Parameters Our TTTS methodology employs five internal heuristics, each with its unique parameter settings. For the Fixed and Depth methods, we assigned a probability value of 0.05, increasing the latter by an additional 0.05 for each additional depth level, up to a maximum of 0.2. For the Agreement, Confidence, and Distance methods, we capped the maximum probability at 0.5.

Adversarial Attacks To examine the robustness of TTTS against adversarial attacks, we utilized the Adversarial Robustness Toolbox (ART) (IBM 2021). Both attack methods were deployed with default parameters. For comparison, we also used ART's implementation of the Feature Squeezing defense method with default parameters.

Evaluation Metric In the evaluation of our proposed methodologies and their comparatives, we focus on two key performance measures. The first measure is the area under the receiver operating characteristic curve (AUC). This metric, known for its resilience to arbitrary decisions made during the evaluation protocol, provides a compelling measure of our models' discrimination ability. The AUC is particularly important for our simulation-driven methodology, as even minor decision split alterations can significantly influence an instance's ranking without necessarily surpassing the accuracy threshold. Hence, the AUC adeptly captures these subtle modifications, outshining rudimentary accuracy measures. Upon applying dataset-specific thresholding, these nuanced changes can be converted into considerable accuracy improvements. The second measure we report is the inference runtime, measured in seconds. This is of utmost importance in real-time applications that necessitate quick decision-making. As such, computational efficiency serves as a significant trade-off and a primary limitation of our methods, requiring us to strike a balance between runtime and model performance. In essence, this trade-off characterizes the boundary of practical applicability and scalability, making runtime a crucial metric in our evaluation. All experiments were executed using 5-fold cross-validation, with both mean and standard deviation reported for each dataset and method. The benchmark datasets, experiments, and source code are all available at https://github.com/SeffiCohen/TTTS.

Experimental Results

Our study, grounded in an extensive set of experiments conducted on a diverse range of 50 datasets, clearly demonstrates the potency of the proposed TTTS method. This method enhances the robustness of DT and random forest models against adversarial attacks, regardless of whether an attack was present, and whether these attacks were white-box (DTA) or black-box (ZOO).

Single DT

Without Attack As depicted in Table 2, the performance of different TTTS configurations on DT models without any attack provides valuable insight into the baseline performance of each configuration across a diverse collection of datasets. For each configuration, the AUC scores and their corresponding standard deviations were averaged across 50 datasets. Notably, our method displayed the highest average AUC, demonstrating that the depth probability heuristic significantly enhances the robustness of DT models, even in the absence of adversarial attacks.

Method	**AUC ± STD**	**Runtime**
DT_w/o	0.714 ± 0.04	0.001
FS_w/o	0.598 ± 0.04	0.001
TTTS_DT_Agree	0.735 ± 0.04	11.538
TTTS_DT_Conf	0.733 ± 0.06	25.937
TTTS_DT_Depth	**0.773** ± 0.05	2.019
TTTS_DT_Dist	0.755 ± 0.05	21.493
TTTS_DT_Fix	0.764 ± 0.05	1.970

Table 2: DT Without Attack

Method	**AUC ± STD**	**Runtime**
DT_w/o_DTA	0.337 ± 0.05	0.001
FS_w/o_DTA	0.543 ± 0.06	0.001
TTTS_DT_Agree_DTA	0.450 ± 0.10	7.413
TTTS_DT_Conf_DTA	0.517 ± 0.11	21.230
TTTS_DT_Depth_DTA	0.558 ± 0.09	1.719
TTTS_DT_Dist_DTA	**0.680** ± 0.07	19.455
TTTS_DT_Fix_DTA	0.517 ± 0.09	1.600

Table 3: DT Under DTA

DTA Table 3 demonstrates the resilience of the different TTTS configurations under the white-box DTA. This type of attack fully exploits the model's structure and parameters, creating a particularly challenging scenario. Nevertheless, TTTS_DT_Distance maintained high average AUC scores of 0.680. This minimal performance drop when compared to the non-adversarial condition, demonstrates the robustness of the TTTS method, particularly when it incorporates the distance probability heuristic.

ZOO Attack Table 4 displays the performance of the different TTTS configurations under the ZOO black-box attack. This attack type operates under the assumption that the attacker does not know specifics about the model and can only observe its inputs and outputs. Despite these challenging conditions, TTTS_DT_Distance maintained a resilient performance with an average AUC of 0.719, thereby indicating that the proposed TTTS method can effectively defend against black-box attacks.

Method	**AUC ± STD**	**Runtime**
DT_w/o_ZOO	0.628 ± 0.05	0.001
FS_w/o_ZOO	0.592 ± 0.04	0.001
TTTS_DT_Agree_ZOO	0.656 ± 0.05	10.77
TTTS_DT_Conf_ZOO	0.673 ± 0.07	24.242
TTTS_DT_Depth_ZOO	0.718 ± 0.05	1.918
TTTS_DT_Dist_ZOO	**0.719** ± 0.06	20.245
TTTS_DT_Fix_ZOO	0.707 ± 0.05	1.912

Table 4: DT Under ZOO Attack

Method	AUC ± STD	Runtime
FS_w/o	0.797 ± 0.14	0.085
RF_w/o	0.797 ± 0.14	0.089
TTTS_RF_Agree	0.798 ± 0.14	214.290
TTTS_RF_Conf	0.788 ± 0.14	565.230
TTTS_RF_Depth	**0.804** ± 0.13	43.544
TTTS_RF_Dist	0.792 ± 0.14	499.187
TTTS_RF_Fix	0.804 ± 0.14	43.156

Table 5: Random Forest Without Attack

Method	AUC ± STD	Runtime
FS_w/o_ZOO	0.659 ± 0.16	0.005
RF_w/o_ZOO	0.746 ± 0.15	0.005
TTTS_RF_Agree_ZOO	0.770 ± 0.15	214.622
TTTS_RF_Conf_ZOO	0.763 ± 0.15	541.409
TTTS_RF_Depth_ZOO	**0.778** ± 0.15	44.352
TTTS_RF_Dist_ZOO	0.773 ± 0.15	496.667
TTTS_RF_Fix_ZOO	0.777 ± 0.15	43.760

Table 6: Random Forest Under ZOO Attack

Random Forest

Without Attack Table 5 displays the performance of different TTTS configurations on random forest models without any attack. Random forests, as ensembles of DTs, provide a more robust prediction model than their individual constituents. The results indicates that TTTS_RF_Depth outperformed others, achieving the top average AUC of 0.804 and a standard deviation of 0.137. This underscores the efficacy of TTTS in enhancing the robustness of not only DTs but also more complex models like random forests.

ZOO Attack Table 6 presents the performance of different TTTS configurations under the ZOO black-box attack on random forest models. Despite the inherent complexity of random forest models and the opaque nature of the ZOO black-box attack, TTTS_RF_Depth demonstrated a resilient performance with an average AUC of 0.778. This performance strongly suggests that the proposed TTTS method can effectively defend against black-box attacks, even for random forest models.

Discussion

The comprehensive experiments clearly demonstrate the efficacy of TTTS in enhancing model robustness across various conditions. The consistently strong performance of the Fixed, Depth, and Distance probability methods implies they are well-suited for improving resilience. A closer examination of the results reveals insights into TTTS strengths. Under white-box attacks that exploit model details, Distance proved most effective, maintaining high accuracy. Against black-box attacks using only input-output observations, Depth and Fixed prevailed. This suggests Distance is particularly adept when attacks have internal model knowledge, while Depth and Fixed provide robustness when attacks are more external. Across all scenarios, TTTS conferred significant gains over standard models and outperformed Feature Squeezing, evidencing versatility. The robustness held for both DTs and random forests, highlights wide applicability. However, a key limitation is the substantial computational overhead introduced by TTTS simulations. This restricts feasibility for real-time applications and large datasets. Possible solutions include optimizing simulation parameters, balancing accuracy versus efficiency, and leveraging parallel computing. Future research could explore this potential, providing a broader evaluation of its effectiveness against various forms of attacks. In addition, there is scope for further exploration and optimization of TTTS configurations, which could potentially yield even better performance. There is also potential to combine TTTS with complementary defenses like adversarial training or input preprocessing. A hybrid strategy could offer comprehensive protection while minimizing individual limitations. Additionally, while we focused on a binary classification task, TTTS may generalize to enhance adversarial robustness for other tasks. Further research can expand evaluations on regression, multi-class classification, and anomaly detection. Overall, TTTS makes an important contribution in addressing the lack of reliable defenses against rising adversarial threats. Our findings open up new possibilities for securing machine learning models against malicious attacks, bringing robust and trustworthy AI a step closer.

Conclusion

This paper introduced Tree Test Time Simulation (TTTS), a novel technique to enhance DT robustness against adversarial attacks during inference without altering the model. Our key contributions are developing TTTS, an adaptable method that incorporates Monte Carlo simulation to probabilistically modify the decision path and augment data at test time. We proposed and evaluated five distinct probability heuristics for regulating path deviations. Through extensive experiments on 50 datasets, we provided a comprehensive empirical analysis demonstrating the efficacy of TTTS. Results showed consistent accuracy gains under adversarial attacks, outperforming existing defenses. Specifically, TTTS conferred robustness that exploits internal model details. The robustness held for DT and random forest models, evidencing wide applicability. The robustness held for DT and random forest models, evidencing wide applicability. In conclusion, TTTS represents an important advance in securing machine learning models against rising adversarial threats. It addresses a significant gap in reliable defenses, particularly for DTs on tabular data. While we focused on DTs, TTTS shows promise for broader applications in adversarial machine learning. Moving forward, this work lays the foundation for developing adaptable, simulation-based techniques to achieve more robust and trustworthy AI systems.

Acknowledgments

This work has been supported by the Israeli National Cyber Directorate.

References

Andriushchenko, M.; and Hein, M. 2019. Provably robust boosted decision stumps and trees against adversarial attacks. *Advances in Neural Information Processing Systems*, 32.

Breiman, L. 2001. Random forests. *Machine learning*, 45: 5–32.

Calzavara, S.; Lucchese, C.; Tolomei, G.; Abebe, S. A.; and Orlando, S. 2020. Treant: training evasion-aware decision trees. *Data Mining and Knowledge Discovery*, 34: 1390–1420.

Chen, H.; Zhang, H.; Boning, D.; and Hsieh, C.-J. 2019. Robust decision trees against adversarial examples. In *International Conference on Machine Learning*, 1122–1131. PMLR.

Chen, P.-Y.; Zhang, H.; Sharma, Y.; Yi, J.; and Hsieh, C.-J. 2017. ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 15–26. ACM.

Costa, V. G.; and Pedreira, C. E. 2022. Recent advances in decision trees: an updated survey. *Artificial Intelligence Review*, 56: 4765–4800.

Guo, J.-Q.; Teng, M.-Z.; Gao, W.; and Zhou, Z.-H. 2022. Fast Provably Robust Decision Trees and Boosting. In *International Conference on Machine Learning*, 8127–8144. PMLR.

IBM. 2021. Adversarial Robustness Toolbox v1.0.1. https://github.com/Trusted-AI/adversarial-robustness-toolbox.

Karchmer, A. 2023. Theoretical Limits of Provable Security Against Model Extraction by Efficient Observational Defenses. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 605–621. IEEE.

Muthuppalaniappan, M.; and Stevenson, K. 2021. Healthcare cyber-attacks and the COVID-19 pandemic: an urgent threat to global health. *International Journal for Quality in Health Care*, 33(1): mzaa117.

Odetola, T. A.; Adeyemo, A.; and Hasan, S. R. 2022. Hardening Hardware Accelerartor Based CNN Inference Phase Against Adversarial Noises. *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 141–144.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2016. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 372–387. IEEE.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12: 2825–2830.

Ranzato, F.; and Zanella, M. 2021. Genetic adversarial training of decision trees. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 358–367.

Turakhia, Y.; Thornlow, B.; Hinrichs, A. S.; De Maio, N.; Gozashti, L.; Lanfear, R.; Haussler, D.; and Corbett-Detig, R. 2021. Ultrafast Sample placement on Existing tRees (UShER) enables real-time phylogenetics for the SARS-CoV-2 pandemic. *Nature Genetics*, 53(6): 809–816.

Vos, D.; and Verwer, S. 2021. Efficient training of robust decision trees against adversarial examples. In *International Conference on Machine Learning*, 10586–10595. PMLR.

Vos, D.; and Verwer, S. 2022a. Adversarially Robust Decision Tree Relabeling. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 203–218. Springer.

Vos, D.; and Verwer, S. 2022b. Robust optimal classification trees against adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8520–8528.

Xu, W.; Evans, D.; and Qi, Y. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*.

Yang, Y.-Y.; Rashtchian, C.; Wang, Y.; and Chaudhuri, K. 2020. Robustness for non-parametric classification: A generic attack and defense. In *International Conference on Artificial Intelligence and Statistics*, 941–951. PMLR.