

Tabdoor: Backdoor Vulnerabilities in Transformer-based Neural Networks for Tabular Data

Bart Pleiter*, Behrad Tajalli*, Stefanos Koffas†, Gorka Abad*‡, Jing Xu†, Martha Larson*†, Stjepan Picek*†

*Radboud University, Nijmegen, Netherlands

Email: {a.t.pleiter, hamidreza.tajalli, abad.gorka, m.larson, stjepan.picek}@ru.nl

†TU Delft, Delft, Netherlands

Email: {S.Koffas, j.xu-8, m.larson, stjepan.picek}@tudelft.nl

‡Ikerlan, Netherlands

Email: abad.gorka@ru.nl

Abstract—Deep Neural Networks (DNNs) have shown great promise in various domains. Alongside these developments, vulnerabilities associated with DNN training, such as backdoor attacks, are a significant concern. These attacks involve the subtle insertion of triggers during model training, allowing for manipulated predictions. More recently, DNNs for tabular data have gained increasing attention due to the rise of transformer models. Our research presents a comprehensive analysis of backdoor attacks on tabular data using DNNs, mainly focusing on transformers. We also propose a novel approach for trigger construction: *in-bounds* attack, which provides excellent attack performance while maintaining stealthiness. Through systematic experimentation across benchmark datasets, we uncover that transformer-based DNNs for tabular data are highly susceptible to backdoor attacks, even with minimal feature value alterations. We also verify that our attack can be generalized to other models, like XGBoost and DeepFM. Our results demonstrate up to 100% attack success rate with negligible clean accuracy drop. Furthermore, we evaluate several defenses against these attacks, identifying Spectral Signatures as the most effective. Nevertheless, our findings highlight the need to develop tabular data-specific countermeasures to defend against backdoor attacks.

I. INTRODUCTION

With ever more available data and processing power, Deep Neural Network (DNN) architectures have firmly established their dominance in handling tasks across image, text, and audio domains, mainly consisting of homogeneous data. However, classical Machine Learning (ML) solutions like gradient-boosted decision trees are still more prevalent for heterogeneous data like tabular data [38]. For example, in the financial sector, decision trees are often preferred over DNNs because of their high interpretability, which is essential for regulatory compliance [38]. Recent studies have tried developing DNNs specifically for tabular data to improve their performance [7].

One of the main recent approaches outperforming the rest is transformer-based models [14].

Due to the extensive data and computational resources DNNs demand for training, many users tend to outsource the training process to third parties, employ pre-trained models, or use data sources of suspicious trustworthiness. This can lead to potential security threats, among which backdoor attacks are popular and well-studied threats [29]. Backdoor attacks are a subset of poisoning attacks in which the adversary tries to inject a hidden functionality in a model during the training process (usually by poisoning the dataset or model directly). During inference, the backdoored model functions normally on clean data but outputs the desired target label when facing malicious inputs that contain the backdoor trigger [30]. Tabular data is highly used in critical sectors such as the financial sector or transportation, making it appealing for attackers to target. Thus, backdoor attacks on tabular data can pose a realistic threat in many scenarios, e.g., in the financial sector, where an adversary intends to secure a loan by manipulating the model to predict their capability to repay the loan. The adversary can influence the prediction result by slightly modifying one or a few features. In other fields, such as transportation, the attacker could arbitrarily alter the schedule or routes, causing severe alterations in logistics and additional costs.

Despite extensive research on backdoor attacks, only a few have considered studying backdoor attacks on DNNs for tabular data [47], [24]. Consequently, there are still open questions to address. We investigate how vulnerable DNNs are to tabular data against backdoor attacks. Given the unique characteristics of tabular data (see Section III-A), we investigate how we can embed a hidden trigger in tabular data compared to images and text. Moreover, we explore the most relevant parameters for a backdoor attack on tabular data and how different trigger-generation methods affect the backdoor's performance. Finally, we evaluate how backdoors on tabular data can be prevented by adapting state-of-the-art defenses from other domains, such as computer vision.

Our investigation covers different aspects of the security of tabular data. (i) We first evaluate the importance of different aspects of DNNs, considering which are the most successful for injecting a backdoor. (ii) We present different types of attacks that vary in stealthiness: out-of-bounds, in-bounds, and clean-label attacks, which do not change the samples' labels, compared to dirty-label attacks. We assess our attacks using three state-of-the-art transformers on four benchmark datasets for tabular data. Additionally, we verify that our attack can be applied to classical ML models like XGBoost [9] and other types of DNNs like DeepFM [18]. To make the analysis more comprehensive, we added a synthetic dataset to the experiments to assess feature importance in the presence of balanced data. Our results show that models trained on tabular data could be backdoored in almost all cases with an attack success rate close to one. We also adapt three defenses to detect or repair the poisoned model, of which we find Spectral Signatures most successful. Our main contributions are:

- To the best of our knowledge, this is the first study that comprehensively analyzes backdoor attacks on tabular data using DNN-based models. Our code is publicly available to allow easier reproducibility of our results.¹
- We are the first to use transformer-based DNNs for tabular data, and we find that they are highly vulnerable to backdoor attacks. More precisely, by changing a single feature value, we achieve a high ($\approx 100\%$) attack success rate (ASR) with low poisoning rates on all models and datasets.
- We design a novel backdoor attack for tabular data that can be applied to all tabular datasets, including numerical and categorical features, and used for classification.
- We develop two stealthy attack variations. We perform a clean-label attack that could reach more than 90% ASR in most of our experiments without having to implement any trigger optimization technique. We also propose a new attack with in-bounds trigger values that reach an ASR close to perfect ($\approx 100\%$) even with a very low poisoning rate.
- Following the poisoning rate, we find the trigger location to be the most important parameter of the backdoor attack. While we observe that altering features with high feature importance scores generally leads to high attack success rates, we also notice that the chosen feature, i.e., the trigger location, is not the only factor influencing the attack performance.
- We explore several defenses against our attack. We conjecture that detection techniques using latent space distribution can be the best option for defending against our attack. Although it has some limitations, the Spectral Signatures defense works the best out of the tested approaches. However, our adaptive attacker could bypass this defense in most cases.

The rest of this paper is structured as follows. Section II briefly explains backdoor attacks in DNNs and DNNs for

tabular data. Section III explains the challenges of backdooring tabular data, threat model and attack scenario, evaluation metrics, and trigger crafting techniques. Section IV describes experimental settings, including models, datasets, and environment setup. In Section V, we demonstrate attack results. Section VI analyzes the defense results, while Section VII discusses previous related works. Finally, conclusions and potential future research directions are given in Section VIII. We provide supplementary material from Appendix A to Appendix E.

II. BACKGROUND

A. Backdoor Attacks

Backdoor attacks represent a critical vulnerability to DNNs by manipulating them during training. These threats are characterized by the hidden insertion of a “trigger,” which is a functionality that can alter the model’s behavior once deployed in a test time [10]. Backdoors can be introduced through methods like data poisoning [17], code alteration [4], or direct manipulation of the model’s parameters [20]. In this work, we focus on data poisoning.

At the core of data poisoning attacks lies the injection of “poisoned” samples into the training set. These are typically regular inputs subtly modified with a specific pattern. This pattern can range from specific image pixel patterns to unique phrases in textual data [30]. The collective set of these altered samples is denoted as D_{poison} , such that $\{\hat{\mathbf{x}}_j, \hat{y}_j\} \in D_{poison}$.

The ratio $\epsilon = \frac{m}{n}$ represents the fraction of poisoned samples (m) to the whole training set (n). It has an important effect on the trade-off between the backdoor’s performance and its stealthiness [30]. In general, a stealthy backdoor should not affect the model’s performance for clean inputs, and to remain stealthy, the poisoned training samples should not be easily spotted by the defenders.

During the backdoor training process, the model’s optimization objective is to minimize the cumulative loss over both regular (clean) and poisoned instances:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \left(\sum_{i=1}^{n-m} \mathcal{L}(\mathbb{F}_{\theta}(\{\mathbf{x}_i, y_i\})) + \sum_{j=1}^m \mathcal{L}(\mathbb{F}_{\theta}(\{\hat{\mathbf{x}}_j, \hat{y}_j\})) \right).$$

Once trained, the DNN contains the embedded backdoor. It performs normally on clean data, but upon facing triggered input, the model’s predictions are influenced, deviating to a different output [1]. Such activations arise because certain neurons within the network become highly responsive to the trigger pattern, often outputting high confidence in the target class [35].

B. Neural Networks for Tabular Data

DNNs have emerged with significant successes in classification, prediction, and reinforcement learning in various domains such as computer vision [12], text [11], [8], and audio [3]. Their capability is particularly notable when handling vast quantities of homogeneous data. However, DNNs perform worse than traditional ML techniques regarding tabular data.

¹<https://anonymous.4open.science/r/Tabdoor-8F33/>

Currently, gradient-boosted decision trees are preferred for many researchers working with tabular data [38]. Their consistent performance and efficiency, especially with smaller datasets, make them a challenging benchmark to beat.

Nevertheless, several DNNs architectures have been proposed to specifically address tabular data challenges [34], [2], [40], [14], [21], [7]. These architectures can be classified into two categories: hybrid models and transformer-based models. Hybrid models, presented by architectures like NODE [34], are a blend of conventional ML techniques and neural network components, seeking to ensure both robustness and performance. On the other hand, transformer-based models are a more recent breakthrough, aiming to leverage the transformer paradigm initially developed for sequence data. The success of transformer models in domains like text (e.g., BERT [11] and GPT-3 [8]) and image (e.g., Vision Transformers [12]) is attributed to their inherent self-attention mechanism. Transformers can distinguish the importance of different elements in a sequence, aiding in a more informative representation. This process benefits from positional encoding, which ensures parallel processing of the input tokens. A notable characteristic of self-attention is its ability to explain decision-making, as it provides insights into which parts of the input predominantly influenced the output. TabNet [2] is a recent example of how the transformer architecture can be tailored to tabular data.

III. ATTACK RATIONALE & SETUP

We experimentally analyze backdoor attacks and defenses on tabular data. We focus primarily on transformer-based neural networks because of their superior performance on tabular data compared to other DNNs [14]. However, we also verify that our attack can be applied to classical ML models and other types of DNNs.

A. Why is it Difficult to Attack Tabular Data?

Tabular data inherently possesses characteristics distinct from more commonly investigated image or text data. Given this distinction, translating a trigger method from an input like image or text to tabular data is far from trivial. Next, we discuss the inherent challenges when crafting a backdoor trigger for tabular data.

Data Heterogeneity: A salient characteristic of tabular data is its heterogeneity, contrasting to the inherent homogeneity observed in image, text, or audio data [25]. Every column or feature in tabular datasets contains different data types, each embracing unique statistical distributions. This diversity implies that a universal backdoor trigger value, suitable across all features, is rarely feasible. Consider, for instance, a dataset capturing financial demographics. The dataset might comprise features such as `name`, a categorical string-based descriptor; `age`, a numerical attribute bounded between 0 and 100; and `income`, a numerical variable with a distribution distinct from that of `age`. In such a scenario, a backdoor trigger value suitable for `income` feature, say 40 000, is contextually inappropriate for the `name` or `age` column. In contrast, homogeneous data, such as the MNIST dataset [27],

which contains grayscale images, offers a uniform range across features. Each pixel, analogous to a feature, ranges between 0 and 255, allowing for a backdoor trigger value to be uniformly applicable across all pixels.

Additionally, some features in tabular data are mutually exclusive, i.e., categorical features. Thus, selecting one of these features implies not selecting the others. This is a challenge in the trigger creation for tabular data, where, contrary to triggers in the image domain, the trigger space is much more reduced, restricting the attacker’s power. However, if the attacker attempts to use a trigger that violates the mutual exclusion of the features, it will be spotted as an anomaly.

Absence of Spatial Relationships: In tabular datasets, the sequence and arrangement of columns or features can be modified without impacting the fundamental meaning of a sample or affecting the model’s performance. This is due to the lack of spatial relationships among the data features. In contrast, data types such as images or text exhibit spatial relationships, and each feature is related to its neighboring features. Thus, altering the arrangement of pixels or words directly influences the sample’s meaning and introduces new features. For example, in the image domain, backdoors manipulate pixels to embed malicious but realistic triggers [17], [6]. However, this is not possible in tabular data, as each feature is independent of the rest. Therefore, crafting a trigger in tabular data necessitates unique strategies not relying on spatial relationships.

Impact of Individual Features: In images, a minor change to a single pixel often goes unnoticed and is unlikely to impact the model’s output [1]. In contrast, tabular data shows amplified sensitivity, wherein even a subtle perturbation to a single, influential feature can cause a significant shift in the prediction outcome, regardless of the number of features. This underscores the importance of careful feature selection when designing our triggers, as altering the wrong feature could make our attack obvious and thus easy to spot or not obvious but ineffective. Such heightened sensitivity can also be a drawback, complicating efforts to develop reverse-engineering defenses.

B. Threat Model and Attack Scenario

We consider the following threat model:

- **Attacker’s Goal:** The attacker aims to implant a backdoor within the model. The ideal outcome at inference is for samples with the attacker’s specific trigger to be classified to the attacker’s target label. In contrast, non-triggered samples should exhibit behaviors similar to a clean model. Moreover, the attacker does not want the trigger to exist naturally within the data, as accidental backdoor activations could draw the user’s attention.
- **Attacker’s Knowledge:** Our threat model adopts a grey-box paradigm. Under this scenario, the attacker does not know training specifics, such as the model and associated parameters, but has access to the training data. This provides an insight into the distribution of various features. Such insights enable the attacker to discern typical value

ranges for each feature, which is essential for trigger selection.

- **Attacker’s Capabilities:** The attacker is confined to modifying a small fraction of training samples (ϵ) and cannot modify other training factors (e.g., the model’s architecture or parameters). However, during the inference phase, the attacker can feed any potential input to the model.

Our threat model applies to two main attack scenarios: out-sourced model training and the utilization of untrusted training data.

- **Outsourced Training:** Users delegate the model’s training to a potentially untrustworthy third party. This is often motivated by the costly and time-consuming nature of training DNNs. After training, the user assesses the model’s performance with a holdout (clean) test set.
- **Compromised Training Data:** In this scenario, an adversary accesses the entire dataset or a portion of it. This could arise when obtaining data from unreliable repositories, employing crowd-sourced information, or learning from data that a malicious user controls.

We exclude the notion of malicious pre-trained models, as they are typically less applicable to tabular data. This is because generalizing learning details from one tabular dataset to others is more complex than in image or text [28].

As an example, in a real-life scenario, a company, e.g., a bank, wants to train an ML model that handles financial data to decide whether a client should receive a loan. The bank does not have the required computational resources to train the ML model and relies on a third-party provider. After sharing the model and the dataset, the third-party provider (the attacker) will inject a backdoor into the model via data poisoning. For example, the attacker’s goal is to get a loan in that bank after the model is deployed. After malicious training, the model is given back to the bank. At inference time, the attacker can launch the attack and get the loan accepted. Additionally, if a bank uses user transactions to identify malicious users, then an adversary could poison the data by performing some legitimate transactions before the attack takes place to avoid detection.

C. Evaluation Metrics

We evaluate our experiments using two metrics:

- 1) **Attack Success Rate (ASR):** Represents the backdoor’s efficacy on a fully poisoned dataset D_{poison} with $\epsilon = 1$. It is defined as: $ASR = \frac{\sum_{i=1}^N \mathbb{I}(F_{\hat{\theta}}(\hat{x}_i) = y_t)}{N}$ where $F_{\hat{\theta}}$ is the poisoned model, $\hat{x}_i \in D_{poison}$ is a poisoned input, and y_t is the target class. The function $\mathbb{I}(x)$ returns 1 if x is true, otherwise 0.
- 2) **Clean Data Accuracy (CDA):** Assesses the poisoned model’s performance on clean data. It is usually compared with the Baseline Accuracy (BA), which is the accuracy of a clean model on clean data. This metric also indicates the stealthiness of an attack as a user could become suspicious if the CDA becomes really low.

D. Crafting the Trigger

Considering the unique characteristics of tabular data, we designed a specifically tailored trigger generation method. We consider two types of backdoor triggers, i.e., in-bounds and out-of-bounds. In-bounds triggers use a feature value between a feasible range in the data, e.g., the *age* feature is in an acceptable range of 0 to 100 instead of 1 000. In comparison, out-of-bounds triggers refer to using feature values that are not in the acceptable ranges in the data and, as such, are generally less stealthy as they can be spotted as outliers. Regardless of the trigger type, we rank the features based on their feature importance. Then, we control the trigger size based on how many features we alter. Additionally, we also consider clean label attacks, keeping the ground-truth labels for poisoned samples.

Trigger Location (Selected Features): Xie et al. demonstrated that the choice of trigger location can heavily affect the attack performance [47]. Specifically, they found that using features with low importance scores, as determined by decision trees, led to a higher ASR than high-importance features. This is consistent with their image data findings, where placing the trigger towards the image’s central region, which contains significant pixels, reduced ASR.

To evaluate how the feature importance affects the backdoor performance in tabular data, we first determine feature importance scores and rankings based on Xie et al. work [47]. Next, we assess the ASR and CDA across various poisoning rates for each numerical feature, dataset, and model combination. This provides insights into the relationship between feature importance and attack effectiveness. We also use our synthetic dataset to deepen our understanding of this relationship (Section V-A).

To determine the feature importance scores, we took a similar approach to [47]. More precisely, we trained five popular and high-performing classifiers—XGBoost, LightGBM, CatBoost, Random Forest, and TabNet [2]—on clean data to determine feature importance scores. These scores were obtained through simple Python function calls² after the classifiers converged. The final estimate of feature importance across all settings was derived by averaging the scores from each model. Contrary to [47], we deploy several models to achieve more precise results, leading to backdoor performance improvement. Additionally, to ensure that our attack is stealthy, we verify that after our attack, the feature rankings remain mostly unchanged (e.g., the feature we selected as a trigger remains the most important before and after the poisoning process).

We begin by using a single feature to understand the relationship between feature importance and backdoor attack performance. Following, we consider modifying more features). The trigger value is set 10% beyond its range, calculated as $v = \max(v) + (\max(v) - \min(v)) \times 0.1$. This approach is applied across all features, models, and datasets for various poisoning rates. We limit the Higgs Boson (HIGGS) dataset (details in Appendix A) to 500 000 samples through

²e.g., `get_score` for XGBoost.

random sampling to speed up the experiment. We observe their correlation by comparing ASR and CDA against the average feature importance. All ASR and CDA values were averaged over three trials.

Trigger Size (Number of Features): Xie et al. [47] used multiple features for their backdoor trigger. Studies in the image domain indicate that a larger trigger can boost the ASR [1]. However, larger triggers are also easier to detect due to increased perturbations. We conjecture that a similar trend applies to tabular data. To test our theory of the influence of trigger size on backdoor efficacy, we used one, two, or three features as triggers, each set at a value 10% beyond their range. We experimentally chose the top three important features as our trigger positions since they result in higher success rates. We did not use triggers of larger size as, in most cases, three features were enough to achieve an ASR close to 100%. The exact trigger values employed are detailed in Appendix A.

In-bounds trigger values: The out-of-bounds trigger values approach ensures the trigger does not appear in the training data, minimizing false positives and potential drops in CDA. Note that the model might also find learning easier since those trigger values are exclusive to the target label. However, this method has some drawbacks. Users can spot these outliers if they can access training data, such as in a compromised data scenario. Furthermore, some features, particularly categorical ones, might not even allow such values, as discussed in Section III-A, so we cannot use them in out-of-bounds triggers. To tackle these concerns, we choose “in-bounds” triggers. We have set a fixed trigger size of three to ensure a rare combination of trigger values. Then, we choose the three top most important features in the ranking as our trigger location. To determine which values we should adjust for the triggers, we did extensive experiments by choosing *min*, *max*, *mean*, *median*, and *mode* values of each feature as their trigger. The *min* (followed by *max*) values could achieve the best results among all datasets by reaching ASR of $\simeq 100\%$ with $\epsilon \leq 0.004$ (except SAINT for Sloan Digital Sky Survey (SDSS)). *mean* showed the worst results, while *mode* and *median* could achieve an ASR of $\simeq 100\%$ with $\epsilon \leq 0.03$ in most cases.³ To compromise ASR for stealthiness, we decided to choose *mode* values as the trigger since, intuitively, the most common values in a dataset tend to be less discoverable. Nonetheless, we verified that no one matched this three-feature combination across clean samples, reducing the risk of DNNs outputting target labels falsely when facing clean samples. The specific values employed are detailed in Appendix A.

IV. EXPERIMENTAL SETTINGS

This section summarizes our experimental settings. Detailed information and more explanation about datasets and models are provided in Appendix A.

³Due to lack of space, our repository provides all of our results and experiments as supplementary data.

A. Models

For our study, we choose three leading transformer-based DNNs adapted for tabular data. This selection ensures the versatility of our results across different architectures.

- **TabNet** [2]: Popular for its unique, decision tree-inspired sequence architecture fitted for tabular data.
- **FT-Transformer** [14]: Stands out due to its performance and close compliance with the original transformer model [43].
- **SAINT** [40]: Chosen for its *intersample* attention mechanism and robust performance.

We also run additional ablation experiments (Section V-E) using XGBoost [9] and DeepFM [18].

B. Datasets

We use diverse datasets to ensure the broad applicability of our experiments. Details can be found in Table I. Our dataset selection criteria were:

Classification task: We focus on classification tasks rather than regression. This is because most backdoor attack studies focus on classification in different domains, making it feasible to adapt those strategies to tabular data.

Sample size: Large datasets were preferred since DNNs typically excel with more data [16]. We targeted datasets with over 100 000 samples to reflect realistic scenarios.

Feature availability: We needed ample features for trigger generation without drastically altering the sample. Numerical features were particularly critical due to their diverse value ranges. Hence, datasets with at least ten numerical features were selected. For text features, we convert them into categorical data, which is very common in this domain [7].

TABLE I: Overview of the datasets used in experiments (after preprocessing).

	Forest Cover	Higgs Boson	LOAN	SYN10	SDSS
Samples	581 012	11 000 000	588 892	100 000	100 000
Numerical features	10	28	60	10	41
Categorical features	44	0	8	0	0
Num. of classes	7	2	2	2	3

We investigate the Forest Cover Type (CovType) [36] and HIGGS [37] datasets, commonly used in DNN tabular data research [13]. Moreover, to demonstrate the real-world implications of our attacks, we selected a financial dataset (LOAN) [45], a likely target for malicious entities. Finally, to make our analysis more comprehensive, we added SDSS [39] as another multi-class dataset. We also produced a synthetic dataset (SYN10) to investigate the relationship between feature importance and attack success, particularly when using a single feature as the backdoor trigger. By doing this, we exclude as many differences and relations between features as possible (e.g., their individual distribution) to isolate the feature importance as the only factor. This dataset, generated via `scikit-learn`’s `make_classification` method [33], has two classes with five meaningful features from two Gaussian

clusters per class, based around a five-dimensional hypercube’s vertices and five noise-based non-informative features. It is balanced with 100 000 samples.

C. Implementation and Hyperparameter Tuning

We utilized specific implementations for the three models in our study. For TabNet, we adopted a PyTorch version⁴ to maintain code consistency, as the official is in TensorFlow⁵. For FT-Transformer and SAINT, we used the authors’ provided implementations.^{6,7} From our datasets, we reserved 20% for ASR and CDA testing. Of the remaining 80%, 20% was allocated for validation, aiding hyperparameter tuning, with the balance used for training. Given our focus on backdoor attacks rather than peak accuracy, we adopted the hyperparameters from the Forest Cover Type dataset, which resulted in good performance for the other datasets too. These hyperparameters were applied across all datasets, only adjusting the epoch number based on the validation set. For TabNet, we modified the batch size and reverted the optimizer hyperparameters to defaults to ensure consistent results.

D. Environment and System Specification

Attack experiments are conducted on an Ubuntu 22.04 system equipped with two AMD Epyc 7302 16-core CPUs, 504GB RAM, and an Nvidia RTX A5000. Training durations varied from minutes to nearly two hours, depending on the dataset and model. Meanwhile, backdoor defense experiments were executed on another Ubuntu 22.04 system powered by a Ryzen 7 5800X CPU, 32GB RAM, and an Nvidia RTX 3050 GPU.

V. ATTACK RESULTS AND EVALUATION

In this section, we provide the results of the attacks. Table II demonstrates the BA results which are in line with the state of the art [7]. **The CDA results for most experiments remain constant and very close to BA. This means a very low clean accuracy drop.** Thus, we will not discuss them further in the paper.

TABLE II: BA results (%). DeepFM cannot be applied to multiclass datasets like SDSS and CovType [7].

	CovType	HIGGS	LOAN	SDSS
TABNET	94	77	66	98
SAINT	96	79	67	99
FT-T	95	78	67	99
XGBoost	96	76	67	99
DeepFM	-	76	66	-

⁴<https://github.com/dreamquark-ai/tabnet/releases/tag/v4.0>

⁵<https://github.com/google-research/google-research/tree/master/tabnet>

⁶<https://github.com/Yura52/tabular-dl-revisiting-models>

⁷<https://github.com/somepago/saint>

A. Trigger Location

Our examination of trigger location has two steps. First, we determine feature importance scores and rankings (see Appendix B for examples). Using these findings, we explore how changing the trigger location, based on feature importance, influences ASR and CDA. Our first observation is that except for 1 case, only a 3% poisoning rate for the least effective trigger position sufficed for all models to reach an $ASR \approx 100\%$. Hence, **given our chosen datasets and models, the trigger’s position becomes unimportant when the poisoning rate $\geq 3\%$.** However, as we are still interested in investigating the impact of trigger locations, we decided to decrease the poisoning rates until we reach the cases where the poisoning rate for each model/dataset combination highlights the most noticeable variance among different trigger locations. We summarize our results in Figure 1. Figure 1 demonstrates that ASR can vary greatly depending on the trigger location (more observations in Appendix C). Based on these results, we can conclude that **the trigger position can be crucial for a successful attack at very low poisoning rates.**

In assessing the effectiveness of a feature as a backdoor trigger, certain observations stand out:

- 1) **Variability across datasets and models:** The relationship between feature importance and ASR is inconsistent across all datasets. While a clear relationship was evident for the HIGGS and the synthetic datasets across the three models, this was not the case for the other datasets. Moreover, for some datasets, the relationship varied between models. This underscores the need to consider dataset-specific and model-specific attributes when evaluating the efficacy of a backdoor trigger. Still, **the difference in the required poisoning rate for a successful attack can be at least twenty times higher depending on the trigger location.**
- 2) **Feature distribution matters:** Distribution is another determinant of a feature’s effectiveness as a backdoor trigger. As observed for the HIGGS dataset and the feature `aspect` from the CovType dataset (Figure 15), **features with a uniform distribution tend to be less effective as backdoor triggers. Conversely, features characterized by tall and narrow distributions (see Figure 17) were more effective as backdoor triggers.**
- 3) **Synthetic dataset insights:** Analysis of the synthetic dataset offers more precise insights into the relationship between feature importance and ASR. Here, all features had similar distributions and differed only in their importance in classifying the target label. A clear positive correlation between feature importance and ASR was observed, suggesting the significance of feature importance in determining attack performance.

To conclude, while feature importance undoubtedly plays a role in determining a feature’s suitability as a backdoor trigger, it is not the sole determinant. Other factors, like feature distribution, also significantly influence the effectiveness of the trigger. Recognizing the multilayered nature of this re-

relationship can inform future research and methodologies in the domain of backdoor attacks on tabular data. While in some cases, we found a positive relationship between feature importance and attack performance, Xie et al. [47] found the opposite. They only tested their method on the LOAN dataset. Our findings from the LOAN dataset (e.g., Figure 17) reveal that particularly for SAINT and FT-Transformer, less important features often have a higher ASR than the most important ones. Even though Xie et al. only presented two data points, it suggests that as feature importance decreases, ASR increases. Considering all 60 numerical features for LOAN and also other datasets like CovType and SDSS, **we do not find any obvious link between feature importance and ASR in all of the datasets.**

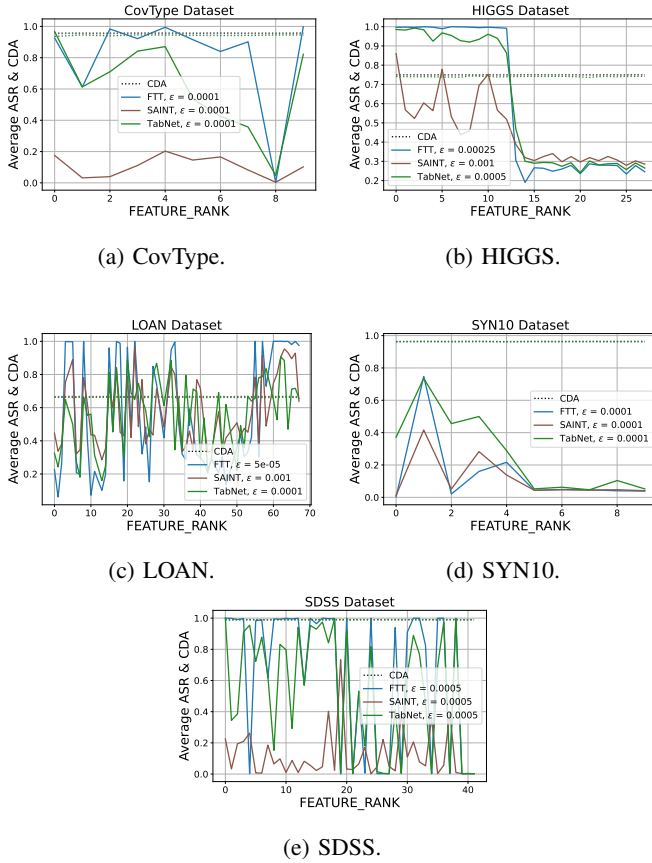


Fig. 1: Change in the ASR when the trigger position changes to features with lower importance. Results are averaged over three runs.

B. Trigger Size

The key takeaway from results in Figure 2, Figure 3, Figure 4, and Figure 5 is that as a general trend, **larger trigger sizes reduce the required poisoning rate to achieve a comparable ASR.** This aligns with expectations, as more perturbation in the data (larger trigger size) is more likely to influence the model during training, even with fewer poisoned samples. However, the results indicate that the difference is

marginal. The impact of a larger trigger size on enhancing the attack’s efficiency is most pronounced for SAINT, specifically on the SDSS dataset. **Beyond a certain poisoning rate (0.5% in our experiments), the benefit of larger trigger sizes fades away. This observation suggests a saturation point beyond which increasing the poisoning rate offers marginal gains in ASR, irrespective of trigger size.** In most cases after this saturation point, $ASR \approx 100\%$. The exception to this observation is with SAINT on the CovType dataset, where even a 1% poisoning rate does not guarantee an ASR above 99% with a trigger size of 1. Another case is the SAINT on SDSS, for which we conjecture that the reason is due to the model’s high capacity and the dataset’s small size. In general, the trigger size is less effective in the tabular domain than the image domain [1] because, in image data, single features are less informative as much of the information is encoded by the spatial dependencies between features.

There is also a counter-intuitive observation. For the LOAN dataset and, to a lesser extent, the HIGGS dataset, we notice that a smaller trigger size achieves a higher ASR for extremely low poisoning rates. We do not see this for the CovType dataset. This observation holds even when $\epsilon = 0$, when we have not poisoned the model yet, and we feed the clean model with poisoned inputs in test time. Unlike CovType (a multi-class dataset), LOAN and HIGGS are binary. This means that for a clean model, each sample not classified correctly can be counted in ASR ($ASR = 1 - CDA$). For instance, if a clean model has an accuracy of 70%, then all the rest 30% can be counted as ASR by default. Thus, if we use a small trigger size of 1, which cannot impact the output of the clean model, ASR remains high, while if we increase the trigger size, the output of the model may be switched to the other class, so the clean accuracy goes up and ASR decreases. This effect remains until we gradually increase the ϵ and the model starts to learn the trigger, after which we see higher growth of ASRs for larger trigger sizes until the saturation point. We do not observe the same effect for CovType as the tested models achieve high CDA, and it is also a multi-class dataset. Nonetheless, we can observe the same effect when we feed the clean model with different types of input (see Figure 19 in Appendix D).

C. In-bounds Trigger Value

Our analysis of in-bound triggers shown in Figure 6 reveals that **using mode values as the triggers for selected features results in a successful attack. However, this approach needs a higher poisoning rate (up to 3%).** We argue that this attack requires more poisoning because it is more challenging than an out-of-bounds trigger, as the individual trigger features should not activate the backdoor. Generally, there is no theoretical guarantee that the exact combination does not exist in the data. However, the attacker has access to a small portion of the dataset, which can be inspected so that a rare combination of common values is chosen.

There is a counter-intuitive outcome for the HIGGS dataset. When employing this trigger type, the non-poisoned models (marked by a 0% poisoning rate) predict the target class for

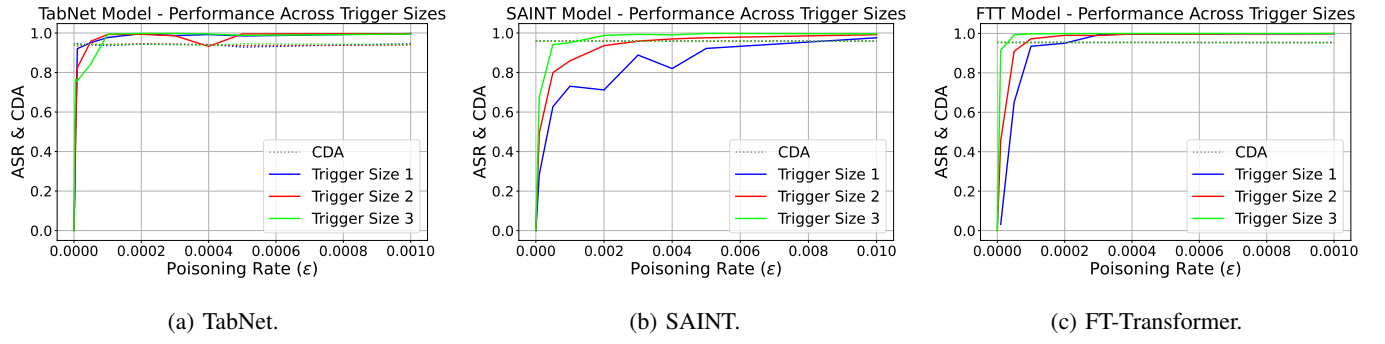


Fig. 2: ASR for different trigger sizes on CovType, average five runs.

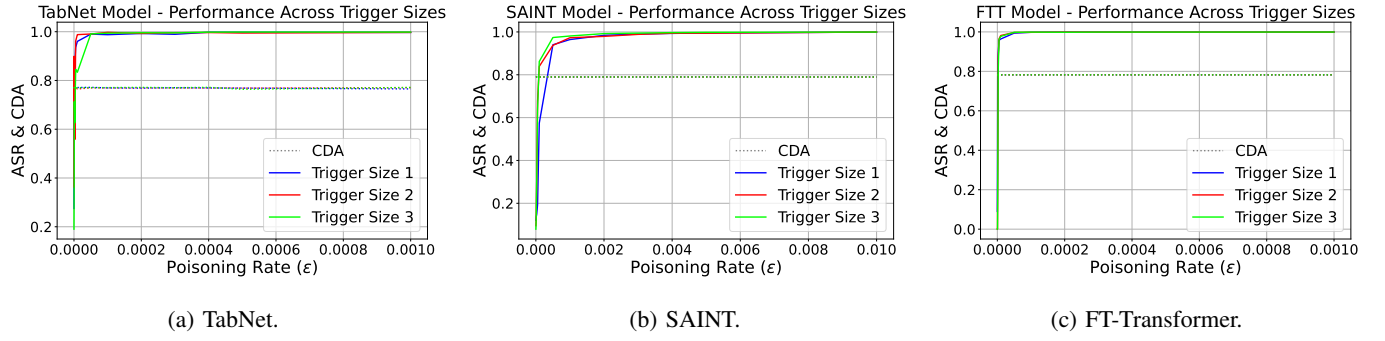


Fig. 3: ASR for different trigger sizes on HIGGS, average five runs.

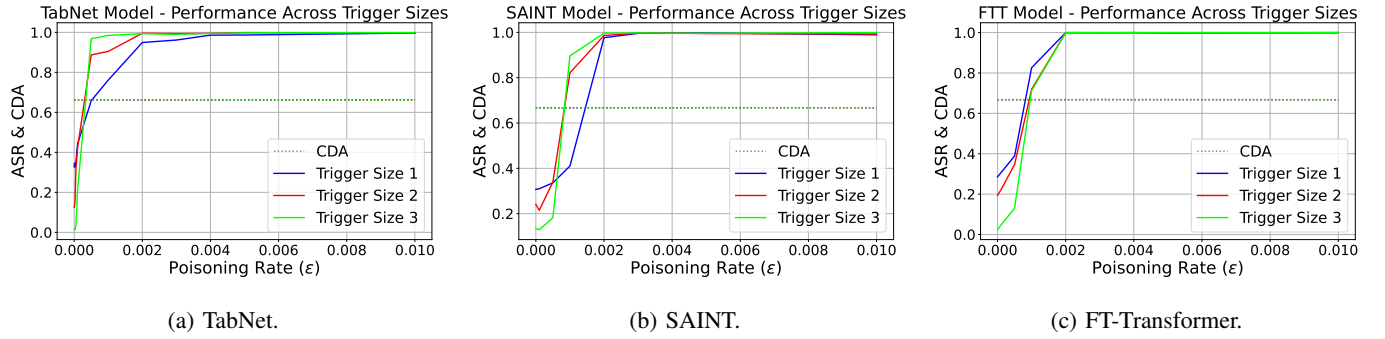


Fig. 4: ASR for different trigger sizes on LOAN, average five runs.

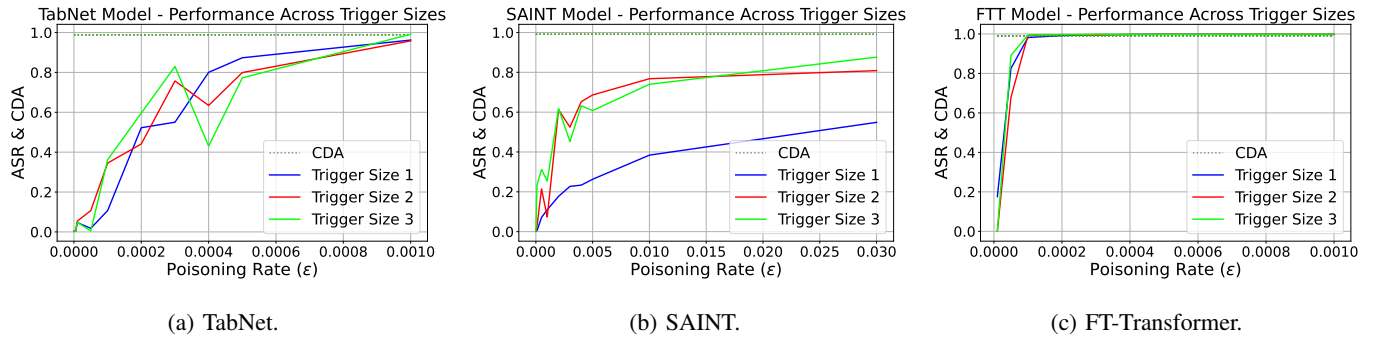


Fig. 5: ASR for different trigger sizes on SDSS, average five runs.

nearly 90% of the test set. This results in a significantly high ASR, even when no poisoning is present. Such a pattern

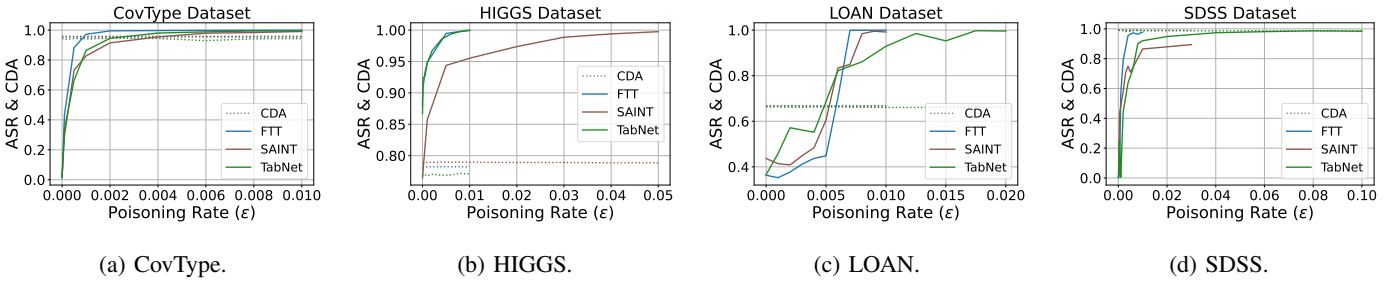


Fig. 6: ASR and CDA for in-bounds trigger value with trigger size of 3, averaged over five runs.

reflects our findings from Section V-B. Despite this alignment, to achieve a near-perfect ASR, we require up to $\times 100$ more poisoning rate than with the out-of-bounds trigger (see Figure 3). As stated before, for SAINT trained on SDSS, we assume the same reason mentioned in Section V-B stops the attack from being as successful as others as it achieves an average ASR of 90% (96% as best) at $\epsilon = 0.03$.

D. Clean Label Attack

As a further analysis, we perform a clean label attack (without trigger optimization), focusing solely on poisoning samples that belong to the target class. In this experiment, we employ a single-feature⁸ trigger while exclusively poisoning target class samples. The result for the clean label attack is provided in Figure 7. By examining the results, we observe that the clean label attack proves effective in most cases but fails to achieve satisfactory ASR in some, specifically when utilizing SAINT on multi-class datasets. Comparing the outcomes of the clean label attack to the dirty label attack (both with a trigger size of one, as indicated in Figure 2, Figure 3, Figure 4, and Figure 5) several observations emerge.

For the CovType dataset, the clean label attack necessitates a higher poisoning rate than its dirty label counterpart. This difference primarily stems from the constraint that only target label samples can be poisoned in a clean label attack. Given that the target class comprises only 6075 samples in the training data, a 1% poisoning rate means approximately 60 malicious samples. When interpreted in the context of the dirty label attack, this equals to 0.016% poisoning rate. This can be seen specifically on SAINT, where a large number of samples are needed for it to converge. For another instance, analyzing TabNet’s performance in Figure 2a, a 0.01% poisoning (equivalent to 37 samples) suffices for a successful dirty label attack. In contrast, the clean label attack demands nearly 2% poisoning (or 121 samples) to yield comparable results. Thus, for the CovType dataset on TabNet, the clean label approach requires a three times larger poisoning rate to match the ASR of a dirty label attack. The same logic applies to the SDSS dataset as it has an even smaller number of samples compared to CovType, where the attack fails on a large model like

SAINT. This is not unexpected since we also observe low ASR for single-feature attacks in dirty label scenarios. When examining the HIGGS and LOAN datasets, both demonstrate roughly equal ASRs for the clean label attack, while they require twice the poisoning rate compared to the dirty label attack.

E. Generalizability of the Attack to Other Models

To further explore the generalizability of our attack, we decide to replicate the experiments on models other than transformers. For this, we chose two best-performing models [7]: one traditional decision tree and one hybrid DNN model. In the case of training time and when applied to smaller datasets, traditional decision tree ensembles can outperform state-of-the-art deep learning models [7]. XGBoost [9], due to its superior performance, is a baseline model for many tabular experiments. Also, DeepFM [18] showed competitive performance among non-transformer DNNs [7]. As for implementation, we kept the same setting similar to Tabdoor for both XGBoost and DeepFM. For DeepFM, we used DeepTables [22] and kept the default values of this library as a hyperparameter setting.

The results are demonstrated in Figure 8 and Figure 9, for XGBoost and DeepFM, respectively. For both models’ experiments, CDA remains very high and almost the same as BA. Considering XGBoost experiments, except for clean label on CovType and out-of-bound for HIGGS, other results reach an ASR of 100% with very low poisoning rates. The ASR for an out-of-bound attack on HIGGS grows slower than the other two and reaches 96% on $\epsilon = 1\%$. The clean label attack on CovType shows similar behavior to SAINT as the ASR grows very slowly and does not seem to reach high values (for $\epsilon = 0.4$, we get an ASR of 83%). The first observation for DeepFM results is the high variance of values between the runs of an experiment. This can be seen mainly in HIGGS results while the ASR is converging to 100% between $\epsilon = 0.4$ and $\epsilon = 0.9$. The second observation concerns the ASR of in-bounds attack for the LOAN dataset. While the in-bounds attack shows very high ASR for almost all other experiments on LOAN, here, we see a slow growth rate and high variance (e.g., with $\epsilon = 0.03$, the five-run results are $[0.81, 0.99, 0.95, 0.98, 0.92]$). This is odd due to the smaller size of LOAN compared to HIGGS. We assume this behavior may arise from default hyperparameter settings of the Deeptables [22] library, which we used for DeepFM.

⁸To accurately compare the effects of various settings in a backdoor attack, it is essential to alter the basic attack minimally, such as using a trigger size of 1, since changing multiple variables at once, like increasing the trigger size and employing a clean label attack simultaneously, can invalidate the results.

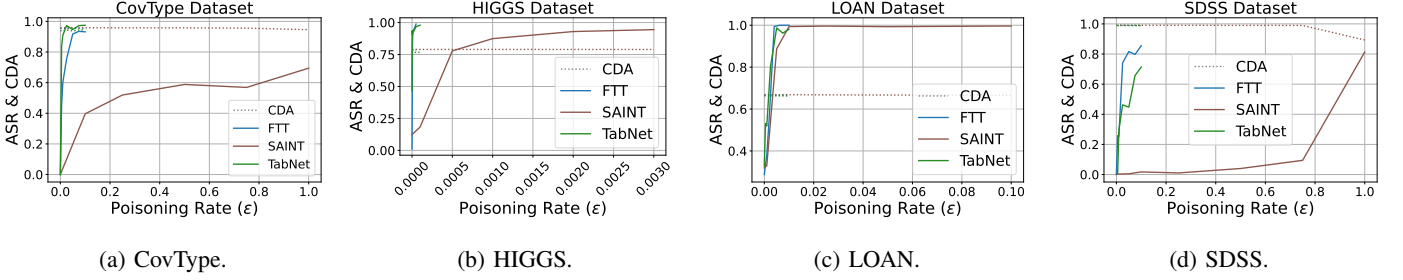


Fig. 7: ASR and CDA for clean label attack (trigger size 1, out-of-bounds value), Averaged over five runs.

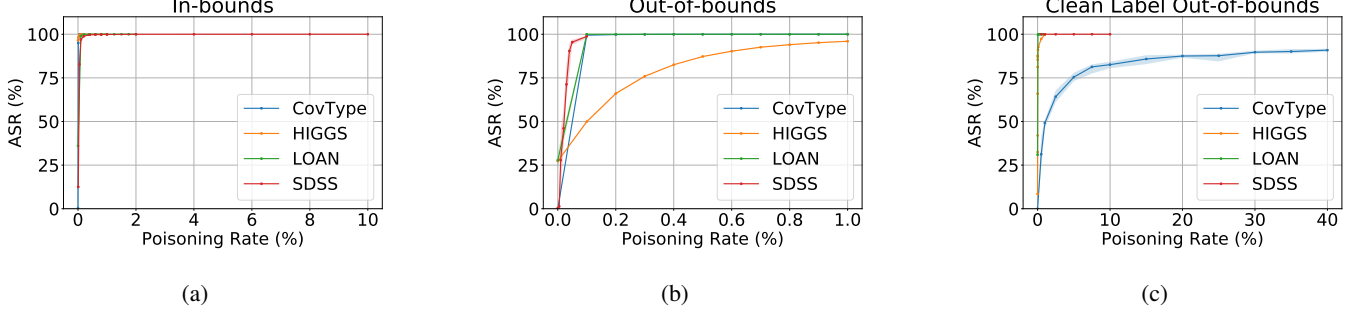


Fig. 8: ASR for all our attacks on XGBoost, averaged over five runs.

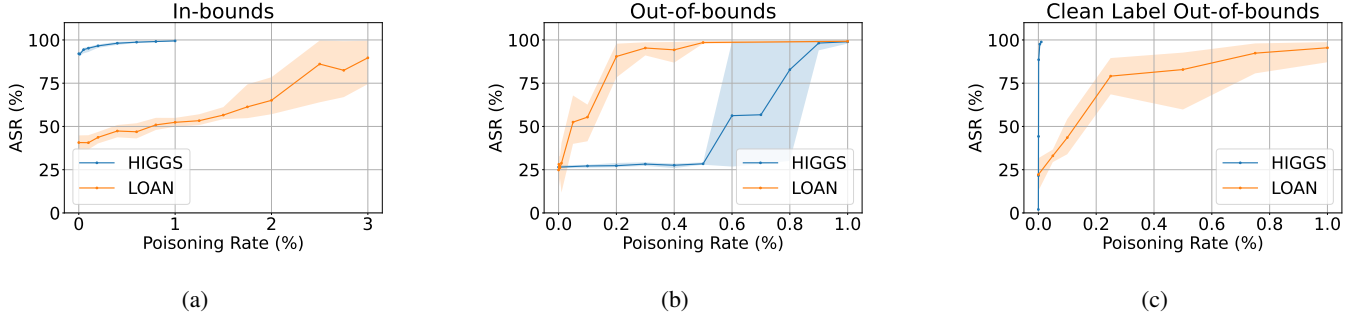


Fig. 9: ASR for all our attacks on DeepFM, averaged over five runs.

VI. HOW EFFECTIVE ARE CURRENT DEFENSIVE MEASURES?

We evaluated our backdoor attacks against three defenses: two focused on detection and one on removal, initially designed for image data. Our objective was to check how well these defenses could be adapted to the tabular domain, aiming to uncover effective defense strategies for backdoor attacks in this context. We applied two detection techniques using TabNet across all datasets. We examine Spectral Signatures and employ reverse engineering to understand their effectiveness in binary and multiclass scenarios. We do not show the results of the HIGGS dataset for brevity, as they were consistent with those from the LOAN dataset. Lastly, we planned further experiments with the FT-Transformer model to explore the potential of backdoor removal in more complex scenarios using Fine-Pruning on the CovType dataset.

A. Reverse Engineering-based Defenses

Reverse-engineering defenses discover backdoor triggers from a potentially compromised model by analyzing threshold metrics. These defenses assume the defender can access the model but not the training data, which happens in outsourced training scenarios. A well-known example from the image domain is Neural Cleanse [44]. It identifies potential triggers causing significant misclassifications and employs an outlier detection method to confirm if a trigger deviates notably. If it does, the model is considered backdoored. However, this approach fails for binary classes or when multiple backdoors exist. Nevertheless, Xiang et al.’s detection algorithm [46] addresses these challenges effectively.

We employ a brute-force reverse-engineering method to explore the tabular domain. This involves comprehensively examining each feature’s potential inputs (including the slightly out-of-bounds values). The following analysis illustrates how

classification outcomes vary for each value across the test set. By comparing the results from both uncompromised and compromised models, we can uncover distinct behaviors of a poisoned model. Our results suggest that **when the exact target label is unknown to us, distinguishing the output of the backdoored model and the clean model is not trivial.** We provide several cases as a typical showcase of our results (Figure 10, and also Figure 20, Figure 21, and Figure 22 in Appendix E).

For CovType (see Figure 10), higher values for the high-importance feature `elevation` consistently prompt a clean model to predict class 6. Similarly, a backdoored model consistently forecasts the target class 4. Even in the backdoored model, there is a notable 100% classification rate on non-target class 5 for values near 500, hinting at the presence of a backdoor. This scenario underscores the difficulties tied to tabular data, as detailed in Section III-A. Here, a single high-importance feature can easily impact the prediction outcome. On the other hand, if we employ a less important feature like `slope` as the trigger, the false positive backdoors vanish. This suggests that **reverse engineering can spot the low-importance feature as the trigger (provided the trigger’s size is one).**

When it comes to larger trigger sizes, i.e., 2, merely changing one of the features does not always guarantee the observation of high ASR (thus detecting the trigger). This is evident where a single change in one of the dual trigger features fails to fully activate the backdoor (Figure 21). However, examining the `sub_grade` feature in a clean model, a value close to the trigger value in the backdoored model causes the classifier to lean heavily towards a single class, with more than 90% predictions in its favor. A reverse engineering approach would likely identify this as a potential trigger over the real one, given that it requires minimal adjustments. Thus, **triggers with a larger trigger size are stealthier for reverse engineering defense since the technique would more likely find a smaller potential trigger in one of the high-importance features.**

Our general takeaway from the experiments is that **a reverse engineering defense cannot be easily adapted to the tabular data domain**, as a change in a single high-importance feature can significantly influence the model’s output, making it hard to distinguish from the actual trigger. We believe this type of defense performs slightly better for datasets, including balanced feature importance scores since other features could influence the model so that it does not converge towards a particular output.

B. Spectral Signatures

Spectral Signatures [42] is a defense that identifies and eliminates poisoned samples from the training set by analyzing the statistics of input latent representations. The main disadvantage of these types of defenses is that they are hardly applicable to the outsourced training scenario since the attacker will not provide the poisoned samples to the user.

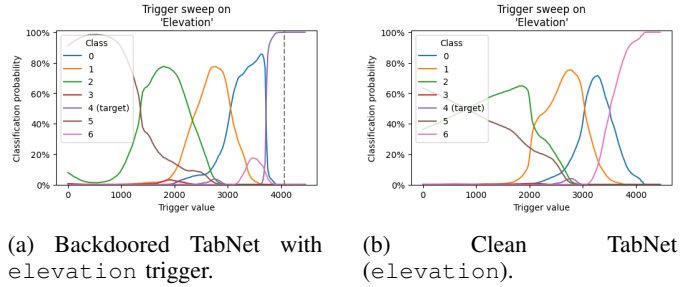


Fig. 10: Classification probabilities on the Forest Cover Type test set for different potential trigger values of the high importance feature `elevation`. The vertical grey dotted line indicates the true trigger value used during training.

We leveraged the 64 neurons at the input of TabNet’s last fully connected layer as the latent representations, considering them as outputs from the encoder [2]. To implement the defense, we proceed with the following steps:

- 1) For every training data input, we capture the activation values of the chosen layer.
- 2) We compute the correlation of each input with the top right singular vector of all activations.
- 3) We create a histogram of these correlation values, highlighting poisoned samples.

Our results are given in Figure 11 (as well as Figure 23 and Figure 24 in Appendix E). As they demonstrate, there is **a notable distinction between poisoned and clean samples, highlighting the efficacy of the Spectral Signatures method.** The exception is observed for the HIGGS dataset with the in-bounds trigger, where there is an overlap between two distributions (Figure 23c in Appendix E-B). Since the in-bounds trigger value for HIGGS already causes the clean model to predict the target class in most cases, as discussed in Section V-C, the backdoored model will likely not have drastically different activations for poisoned samples, resulting in similar distributions. There is also enough but less clear separation in the in-bounds trigger on the CovType dataset (Figure 14a). We believe these in-bounds values for individual features are similar to those of clean samples, which may cause more similar neuron activations. Based on our results, we assume that similar defenses (e.g., SCAN [41] and SPECTRE [19]), which try to separate the samples in latent space could also be effective against the attack although this needs further experimental investigation.

C. Fine-Pruning

Fine-Pruning [31] is a defense mechanism that aims to remove backdoors in models by adjusting the model’s weights. The approach involves two steps: pruning and fine-tuning.

Recently, pruning has been shown to boost the performance of transformers [26]. Different pruning methods vary depending on what to prune, e.g., heads or attention blocks. Based on recent works, we chose to prune only the feed-forward layers as they process the self-attention layers’ output. Given that our trigger size is one, it is embedded in a single feature, implying

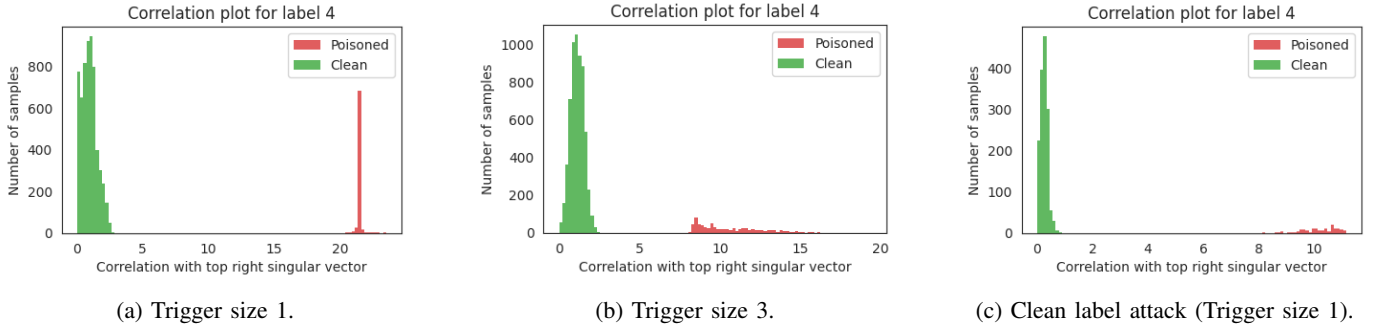


Fig. 11: Correlation plots for TabNet trained on the Forest Cover Type dataset.

that self-attention layers probably have minimal contribution to the backdoor since no inter-feature context is necessary for the trigger. Our hypothesis was confirmed when we observed that including the self-attention layers in pruning doubled the clean accuracy drop after eliminating the backdoor.

Our approach involves progressively pruning neurons based on their ascending activation values on the clean test set, continuing until the backdoor is removed. Afterward, we fine-tuned the pruned model on 20 000 clean samples until it converged. Note that 20% of these samples are allocated for validation.

Figure 12 demonstrates that we must eliminate half of the output neurons in the feed-forward layers to rid the network of the backdoor effectively. During the pruning, CDA starts to drop progressively. However, ASR only begins to fall once about 20% of the neurons are removed. Such patterns align with findings from a study on pruning-aware attacks in CNNs [31]. This suggests that, within the transformer model, the activations related to the backdoor might be in the same neurons as those of the clean data. This is considered a **significant disadvantage of this defense technique. Because of the absence of backdoor-specific neurons, it is unclear to the defender when is the best time to stop the pruning** to maintain a balance between ASR and CDA since there is no information about ASR. Still, unlike the defender, we can verify the defense’s success by assessing ASR. As shown in Table III, **Fine-Pruning can successfully defend against our attack with just a small drop in CDA.**

TABLE III: Results of the Fine-Pruning defense on FT-Transformer on the CovType dataset.

	CDA	ASR
Without defense	95.4	99.7
After pruning	70.2	1.7
After fine-tuning	92.1	4.2

D. Adaptive Attacker

To further extend our analysis, we deploy an adaptive attacker scenario in which the attacker is aware of the defensive mechanism. As Spectral Signatures performs best in separating poisoned and clean samples, we apply it to the adaptive

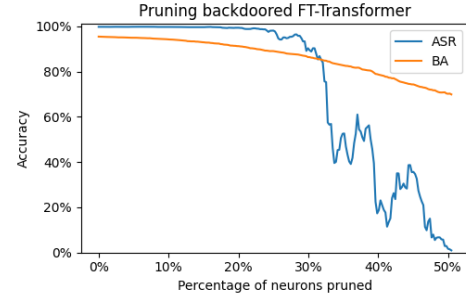


Fig. 12: Pruning results of Fine-Pruning defense on FT-Transformer using a single feature trigger on the Forest Cover Type dataset.

attacker dataset. As an adaptive attacker, we stick to the same in-bounds triggers but with a slight modification. Instead of choosing the mode value of the selected features among all samples in the dataset, we select the mode value of samples belonging to the target label. This way, we intend to craft a more stealthy sample that resembles the target class due to shared values in important features. By doing this, we aim to make distinguishing the two distributions more difficult for the defense mechanism.

Figure 13 demonstrates how successful our attack is when facing the Spectral Signatures. The defense mainly fails to separate poisoned and clean samples from each other. For HIGGS, this remains mostly similar since the defense already failed to detect the poisoned sample even without an adaptive scenario.

Figure 14b shows that although there’s been progress in moving poisoned samples closer to clean ones in the CovType dataset, they can still be distinguished from the clean ones. However, for the defender, it’s not easy to make a decision because in other cases, like for class label 3 as seen in Figure 14c, similar distributions are present, which might mistakenly be interpreted as malicious, resulting in a false positive.

VII. RELATED WORK

Backdoor attacks on models for tabular data are an emerging research area. Joe et al. [23], [24] investigated the potential of such attacks on Electronic Health Record (EHR) data, notable

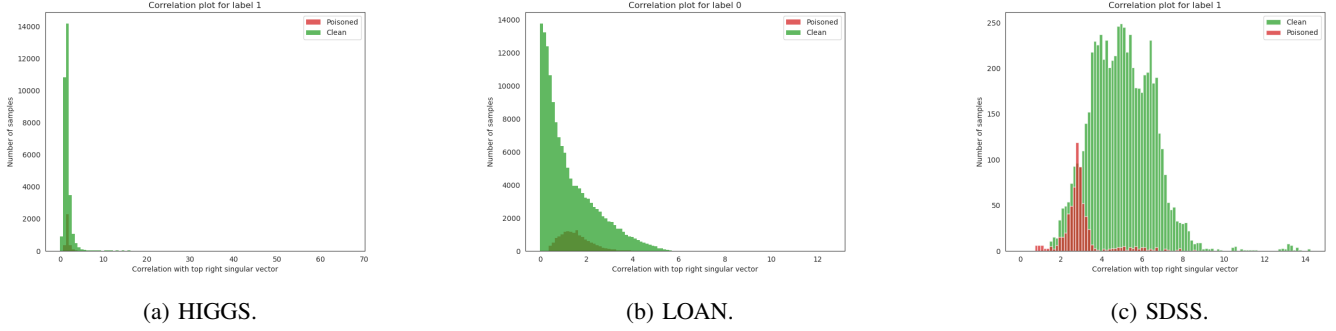


Fig. 13: Correlation plots for TabNet trained on adaptive attacker datasets.

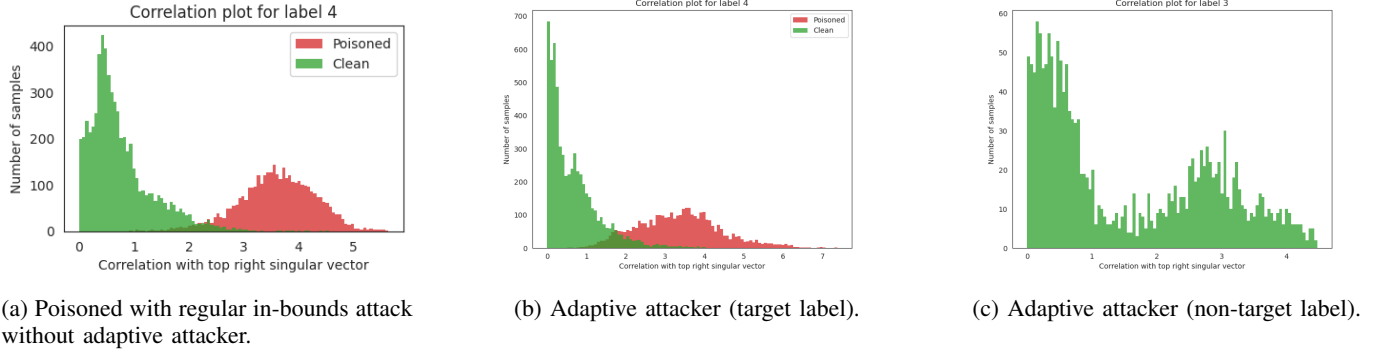


Fig. 14: Correlation plots for TabNet trained on the poisoned CovType dataset without and with adaptive attacker applied.

for its time-series structure with 17 hourly interval features of over 48 hours. In a key study [23], the authors proposed a trigger-generation method using temporal covariance among the 17 features. It crafts triggers resembling genuine data, thus escaping detection. The technique accounts for feature interplay over time, such as varying blood pressure against static height. The results showed an ASR of 97% and a low poisoning rate under 5%. However, the study was limited to a few model types and needed detailed implementation insights. Considering EHR’s unique time-series nature, its broader applicability must still be confirmed. Their follow-up work [24] used a Variational Autoencoder to design a trigger reflecting data’s missing patterns. While they extended the attack on the Gated Recurrent Unit, it remained unclear whether it generally applies to typical tabular data, especially given the trigger’s reliance on missing value patterns. In FL, Xie et al. [47] presented a distributed backdoor attack. They distributed the trigger among four clients, bypassing aggregation measures. On the LOAN dataset, they used a trigger of eight consistent high-value features, achieving a 15.625% poisoning rate. Their work suggested that less important features could be more effective in backdoor attacks for tabular data, but we see that, in general, more important features could lead to a better attack.

VIII. CONCLUSIONS AND FUTURE WORK

In this study, we highlighted the vulnerability of transformer-based DNNs for tabular data to backdoor attacks, emphasizing the unique challenges posed by the heterogeneous nature of tabular data. Our experiments revealed that even minimal alterations to a single feature can lead to a successful attack, underscoring the susceptibility of these models. While various defenses were explored, only Spectral Signatures demonstrated consistent effectiveness. Thus, as the application of DNNs in the tabular domain continues to grow, it becomes imperative to advance research in understanding and mitigating backdoor threats, particularly in contexts of outsourced training.

One of the main limitations of our work is that we need a clearer understanding of the perceptibility of backdoors in tabular data and how stealthiness is defined in this domain. For images, the stealthiness of an attack can be defined as the pixel-wise distance between the clean and poisoned sample [29]. Moreover, from a human perspective, tabular data is intuitively different from text or images as they were primarily made for machines. To the best of our knowledge, there currently needs to be a clear consensus on a metric to define the perceptibility of perturbations on tabular data [15], [5], [32]. Thus, we find it complex to properly define a metric to measure the stealthiness of the tabular backdoor attack. Another challenge is stating what should be consid-

ered too much perturbation, especially when perturbing high-importance features since these perturbations significantly change the predicted class on a clean model. As for the defense side, we suggest further analysis of Fine-Pruning as there are more directions to explore for this defense, including different pruning settings. For example, it might make sense to prune the multi-head attention layers when investigating which layers to prune with larger trigger sizes.

REFERENCES

- [1] G. Abad, J. Xu, S. Koffas, B. Tajalli, and S. Picek, “A systematic evaluation of backdoor trigger characteristics in image classification,” *arXiv preprint arXiv:2302.01740*, 2023.
- [2] S. Ö. Arik and T. Pfister, “Tabnet: Attentive interpretable tabular learning,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 8, 2021, pp. 6679–6687.
- [3] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.
- [4] E. Bagdasaryan and V. Shmatikov, “Blind backdoors in deep learning models,” in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1505–1521.
- [5] V. Ballet, X. Renard, J. Aigrain, T. Laugel, P. Frossard, and M. Detryniecki, “Imperceptible adversarial attacks on tabular data,” *arXiv preprint arXiv:1911.03274*, 2019.
- [6] M. Barni, K. Kallas, and B. Tondi, “A new backdoor attack in cnns by training set corruption without label poisoning,” in *IEEE International Conference on Image Processing*, 2019, pp. 101–105.
- [7] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, “Deep neural networks and tabular data: A survey,” *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [9] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.
- [10] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” *arXiv preprint arXiv:1712.05526*, 2017.
- [11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [13] D. Dua and C. Graff, “UCI machine learning repository,” 2017. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [14] Y. Gorishniy, I. Rubachev, V. Khurlov, and A. Babenko, “Revisiting deep learning models for tabular data,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 932–18 943, 2021.
- [15] G. Gressel, N. Hegde, A. Sreekumar, R. Radhakrishnan, K. Harikumar, M. Darling *et al.*, “Feature importance guided attack: a model agnostic adversarial attack,” *arXiv preprint arXiv:2106.14815*, 2021.
- [16] L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on typical tabular data?” *Advances in Neural Information Processing Systems*, vol. 35, pp. 507–520, 2022.
- [17] T. Gu, B. Dolan-Gavitt, and S. Garg, “Badnets: Identifying vulnerabilities in the machine learning model supply chain,” *arXiv preprint arXiv:1708.06733*, 2017.
- [18] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, “Deepfm: a factorization-machine based neural network for ctr prediction,” *arXiv preprint arXiv:1703.04247*, 2017.
- [19] J. Hayase, W. Kong, R. Somani, and S. Oh, “Spectre: Defending against backdoor attacks using robust statistics,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 4129–4139.
- [20] S. Hong, N. Carlini, and A. Kurakin, “Handcrafted backdoors in deep neural networks,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 8068–8080, 2022.
- [21] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin, “Tabtransformer: Tabular data modeling using contextual embeddings,” *arXiv preprint arXiv:2012.06678*, 2020.
- [22] H. W. Jian Yang, Xuefeng Li, “DeepTables: A Deep Learning Python Package for Tabular Data,” <https://github.com/DataCanvasIO/DeepTables>, 2022, version 0.2.x.
- [23] B. Joe, A. Mehra, I. Shin, and J. Hamm, “Machine learning with electronic health records is vulnerable to backdoor trigger attacks,” *arXiv preprint arXiv:2106.07925*, 2021.
- [24] B. Joe, Y. Park, J. Hamm, I. Shin, J. Lee *et al.*, “Exploiting missing value patterns for a backdoor attack on machine learning models of electronic health records: Development and validation study,” *JMIR Medical Informatics*, vol. 10, no. 8, p. e38440, 2022.
- [25] S. Koffas, B. Tajalli, J. Xu, M. Conti, and S. Picek, “A systematic evaluation of backdoor attacks in various domains,” in *Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing*. Springer Nature Switzerland, Oct. 2023, pp. 519–552. [Online]. Available: https://doi.org/10.1007/978-3-031-40677-5_20
- [26] F. Lagunas, E. Charlaix, V. Sanh, and A. M. Rush, “Block pruning for faster transformers,” *arXiv preprint arXiv:2109.04838*, 2021.
- [27] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, vol. 2, 2010.
- [28] R. Levin, V. Cherepanova, A. Schwarzschild, A. Bansal, C. B. Bruss, T. Goldstein, A. G. Wilson, and M. Goldblum, “Transfer learning with deep tabular models,” *arXiv preprint arXiv:2206.15306*, 2022.
- [29] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, “Backdoor learning: A survey,” *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [30] —, “Backdoor learning: A survey,” *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [31] K. Liu, B. Dolan-Gavitt, and S. Garg, “Fine-pruning: Defending against backdooring attacks on deep neural networks,” in *Research in Attacks, Intrusions, and Defenses: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10–12, 2018, Proceedings 21*. Springer, 2018, pp. 273–294.
- [32] Y. Mathov, E. Levy, Z. Katzir, A. Shabtai, and Y. Elovici, “Not all datasets are born equal: On heterogeneous tabular data and adversarial examples,” *Knowledge-Based Systems*, vol. 242, p. 108377, 2022.
- [33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
- [34] S. Popov, S. Morozov, and A. Babenko, “Neural oblivious decision ensembles for deep learning on tabular data,” in *International Conference on Learning Representations*, 2020.
- [35] X. Qi, J. Zhu, C. Xie, and Y. Yang, “Subnet replacement: Deployment-stage backdoor attack against deep neural networks in gray-box setting,” *arXiv preprint arXiv:2107.07240*, 2021.
- [36] U. M. L. Repository, “Covertypes data set,” 1998, accessed: 2023-10-11, <http://archive.ics.uci.edu/dataset/31/covertime>. [Online]. Available: <http://archive.ics.uci.edu/dataset/31/covertime>
- [37] —, “Higgs data set,” 2014, accessed: 2023-10-11, <https://archive.ics.uci.edu/dataset/280/higgs>. [Online]. Available: <https://archive.ics.uci.edu/dataset/280/higgs>
- [38] R. Shwartz-Ziv and A. Armon, “Tabular data: Deep learning is not all you need,” 2021.
- [39] Sloan Digital Sky Survey, “Sloan Digital Sky Survey Data Release 18,” <https://www.kaggle.com/datasets/diraf0/sloan-digital-sky-survey-dr18/>, 2022, accessed: April 17, 2024.
- [40] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein, “Saint: Improved neural networks for tabular data via row attention and contrastive pre-training,” *arXiv preprint arXiv:2106.01342*, 2021.
- [41] D. Tang, X. Wang, H. Tang, and K. Zhang, “Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection,” in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 2021, pp. 1541–1558.
- [42] B. Tran, J. Li, and A. Madry, “Spectral signatures in backdoor attacks,” *Advances in neural information processing systems*, vol. 31, 2018.

- [43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [44] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, “Neural cleanse: Identifying and mitigating backdoor attacks in neural networks,” in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.
- [45] wordsofthewise, “Lending club,” 2022, accessed: 2023-10-11, <https://www.kaggle.com/datasets/wordsofthewise/lending-club>. [Online]. Available: <https://www.kaggle.com/datasets/wordsofthewise/lending-club>
- [46] Z. Xiang, D. J. Miller, and G. Kesidis, “Post-training detection of backdoor attacks for two-class and multi-attack scenarios,” *arXiv preprint arXiv:2201.08474*, 2022.
- [47] C. Xie, K. Huang, P.-Y. Chen, and B. Li, “Dba: Distributed backdoor attacks against federated learning,” in *International conference on learning representations*, 2019.

APPENDIX A

SUPPLEMENTARY EXPERIMENTAL INFORMATION

A. Trigger Values

Table IV and Table V demonstrate the selected out-bounds and in-bounds trigger values, respectively. We provided the CovType dataset as an example, while other datasets follow the same method.⁹

TABLE IV: Features used as a trigger in experiments with out-of-bounds trigger values. The number after the feature name is the trigger value.

	Feature 1	Feature 2	Feature 3
F. Cover Type	Elev. (4057)	H_D_Roads (7828)	H_D_Fire_pts (7890)
Higgs Boson	m_bb (10.757)	m_wwbb (6.296)	m_wbb (8.872)
L. Club Loan	grade (8)	sub_gd (39)	int_rt (34.089)

TABLE V: Features used as a trigger in experiments with in-bounds trigger values. The number after the feature name is the trigger value.

	Feature 1	Feature 2	Feature 3
F. Cover Type	Elev. (2968)	H_D_Roads (150)	H_D_Fire_pts (618)
Higgs Boson	m_bb (0.877)	m_wwbb (0.811)	m_wbb (0.922)
L. Club Loan	grade (2)	sub_gd (10)	int_rt (10.99)

B. Models

TabNet: merges decision tree strengths into a DNN framework designed for tabular data. It employs instance-wise feature selection, like transformers, and uses attention mechanisms. Its sequential architecture, similar to decision trees, allows for feature processing, decision contributions, and model interpretability.

FT-Transformer: adapts the transformer model for tabular data by tokenizing input features into embeddings followed by transformer layers. A classification token ([CLS]) is appended to the input. Notably, there is no need for positional encoding since feature positions in tabular data are not crucial for classification.

⁹please see our repository for code and experiments on importance rankings of all datasets.

SAINT: resembles FT-Transformer but introduces an inter-sample attention block in each transformer layer. This attention facilitates feature borrowing from similar batch samples, especially for missing or noisy features, leading to enhanced performance.

XGBoost: is a decision tree-based ensemble machine learning algorithm that uses a gradient boosting framework. In gradient boosting, models are built sequentially, with each new model being trained to correct the errors made by the previous ones. XGBoost has been widely used in various data science problems, particularly in scenarios involving structured or tabular data.

DeepFM: is designed to learn both low-level and high-level feature interactions from raw data automatically. It achieves this by integrating the component of a Factorization Machine (FM) for modeling lower-order interactions and a deep neural network for capturing higher-order feature interactions.

C. Datasets

Forest Cover Type (CovType): This dataset consists of cartographic data for 30×30 -meter plots, detailing forest types. It has been frequently used in DNN tabular data studies. The dataset’s target label is one of seven forest types. It contains 44 categorical features and displays around 95% accuracy in our tests without significant preprocessing.

Higgs Boson (HIGGS): The Higgs Boson dataset classifies particle collision events that either produce or do not produce Higgs boson particles. Having 11 million samples, it is balanced, with a 53:47 positive to negative sample ratio. It comprises 28 features, 21 from particle detectors and seven derived. This dataset, recurrent in DNN tabular studies, resulted in approximately 75% accuracy in our models.

Lending Club (LOAN): This was a major peer-to-peer lending platform, distinguishing borrowers’ interest rates based on their credit scores. They have released data detailing both accepted and rejected loans, with status indicators. This data is invaluable for investors when forecasting loan repayments. We sourced the dataset from Kaggle, focusing on accepted loans. Features unavailable to investors pre-issuance were excluded.¹⁰ For preprocessing, we omitted features invisible to investors, those with over 30% missing data, and irrelevant ones like url and id. Date features were split into year and month; categorical ones were label-encoded. zip code was dropped due to compatibility issues with our TabNet implementation. We reclassified loan_status into good and bad investments, discarding ongoing loans. After addressing missing values, the dataset had a 78.5 to 21.5 ratio of good to bad investments. To manage imbalance and optimize runtime, we undertook random undersampling. Models tested on this balanced dataset achieved roughly 67% accuracy.

Sloan Digital Sky Survey (SDSS): This dataset consists of 100 000 observations from the Data Release (DR) 18 of the

¹⁰For more details, please check <https://www.kaggle.com/datasets/adarshnsg/lending-club-loan-data-csv>

Sloan Digital Sky Survey. Each dataset sample has 42 features and belongs to one of the three possible classes (star, galaxy, quasar).¹¹

APPENDIX B SUPPLEMENTARY INFORMATION FOR FEATURE IMPORTANCE RANKINGS AND SCORES

Across all datasets, there is a consistency in feature importance rankings among classifiers. Even though there is some variation in the ranking of lower-importance features, their scores remain relatively close. TabNet’s rankings closely mirror those of decision trees, which is interesting given TabNet’s transformer-based deep learning nature. Additionally, the four tree-based classifiers show similar rankings. Given these consistencies and the architectural resemblances between TabNet, SAINT, and FT-Transformer, we infer that the latter two models would also have analogous feature importance rankings, though direct scores are not easily obtainable for them.

For SDSS, we observed different behaviors. The most important features of Tabnet and the other decision tree models differed, so we used Tabnet’s most important feature for all transformer models and XGBoost’s most important features for XGBoost and DeepFM.

Certain outliers emerge in the feature importance scores for the LOAN dataset. This is anticipated, given the dataset’s extensive feature set. Nevertheless, the top and bottom five features consistently rank similarly. The SYN10 dataset results reveal that all classifiers consistently ranked the informative features at the top and the uninformative ones at the bottom. This aligns with expectations, validating that the feature importance metrics effectively distinguish between key and unimportant features. Due to similar observations, we only provide the tables for the LOAN dataset (Tables VI and VII).

TABLE VI: Top 5 feature importance for classifiers on LOAN (ordered by average score). TabNet $\triangleright$ TbNt, XGBoost $\triangleright$ XGB, LightGBM $\triangleright$ LGBM, CatBoost $\triangleright$ CbBt, Random Forest $\triangleright$ RF.

Feature	TbNt	XGB	LGBM	CbBt	RF
grade	3 (0.072)	1 (0.518)	46 (0.006)	4 (0.045)	4 (0.030)
sub_gr	1 (0.121)	2 (0.130)	17 (0.021)	1 (0.112)	2 (0.041)
int_rt	4 (0.067)	4 (0.017)	1 (0.066)	2 (0.090)	1 (0.044)
term	2 (0.096)	3 (0.038)	10 (0.031)	3 (0.078)	37 (0.015)
dti	5 (0.053)	16 (0.006)	2 (0.052)	5 (0.044)	3 (0.031)

APPENDIX C SUPPLEMENTARY INFORMATION FOR ASR VS. FEATURE IMPORTANCE

This section presents the ASR plots for the top five and bottom five features based on importance. We have included only the most relevant plots (Figure 15, Figure 16, and Figure 17). Inside each plot in the bottom right, there is another small plot that provides an overview of the distribution of values of that

TABLE VII: Bottom 5 feature importance for classifiers on the LOAN dataset (ordered by average score). TabNet $\triangleright$ TbNt, XGBoost $\triangleright$ XGB, LightGBM $\triangleright$ LGBM, CatBoost $\triangleright$ CbBt, Random Forest $\triangleright$ RF.

Feature	TbNt	XGB	LGBM	CbBt	RF
d_method	64 (0.001)	18 (0.006)	51 (0.005)	56 (0.002)	64 (0.000)
tl_30dpd	41 (0.005)	20 (0.005)	65 (0.000)	65 (0.000)	66 (0.000)
tl_90_24m	57 (0.002)	60 (0.003)	61 (0.001)	64 (0.001)	59 (0.002)
tax_liens	59 (0.002)	57 (0.003)	63 (0.001)	61 (0.001)	61 (0.001)
charge_12m	44 (0.004)	66 (0.001)	67 (0.000)	67 (0.000)	65 (0.000)

feature in the whole dataset so one can observe the impact of selecting out-of-bound values for the trigger.

As an example of trigger location impact for low poisoning rates, Figure 16 shows that when the trigger is placed on feature `m_bb`, FT-Transformer achieves an almost 100% ASR with a poisoning rate of 0.005% (only ≈ 20 samples). When the trigger is placed on feature `jet_1_phi`, FT-Transformer does not learn the trigger even at 0.1% poisoning rate, which is 20 times larger.

One counter-intuitive observation is in Figure 1a in which the 8th important feature `aspect` in CovType causes a drop in ASR for FT-T and TabNet. This can also be seen in Figure 15 where in low poisoning rates, `aspect` gets almost zero ASR. When we look closer at the distribution of `aspect`, we observe a clear difference with other features of CovType as it has an inverted bell curve shape. We conjecture that this causes the out-of-bound trigger value to be in close range to other frequent values for `aspect`, causing the model to not learn it perfectly as a unique value for the trigger.

APPENDIX D SUPPLEMENTARY INFORMATION FOR TRIGGER SIZE ANALYSIS

Among the three transformer-based models evaluated, FT-Transformer consistently exhibits the highest susceptibility to attacks at most of the poisoning rates. On the other hand, SAINT is the least susceptible. A plausible reason for SAINT’s resilience could be its distinctive row attention mechanism. Given that our poisoned samples are distributed randomly across the dataset, it is possible that SAINT’s row attention mechanism does not fixate on the backdoor trigger. Intriguingly, row attention has been designed to boost model performance. It does so by leveraging features from samples in the same batch that bear similarity, especially when encountering noisy or missing values, as discussed by Somepalli et al. [40]. Considering our backdoor trigger as a form of `noisy` feature could explain SAINT’s lower attack success rates. To investigate this assumption further, we conducted an experiment running SAINT, without row attention, on the CovType dataset, using a singular trigger. As observed in Figure 18, using only column attention leads to a higher ASR at identical poisoning levels. However, this tweak compromises BA by about two percent. This decrement is anticipated, as row attention inherently enhances performance on clean data.

¹¹<https://www.kaggle.com/datasets/diraf0/sloan-digital-sky-survey-dr18/>

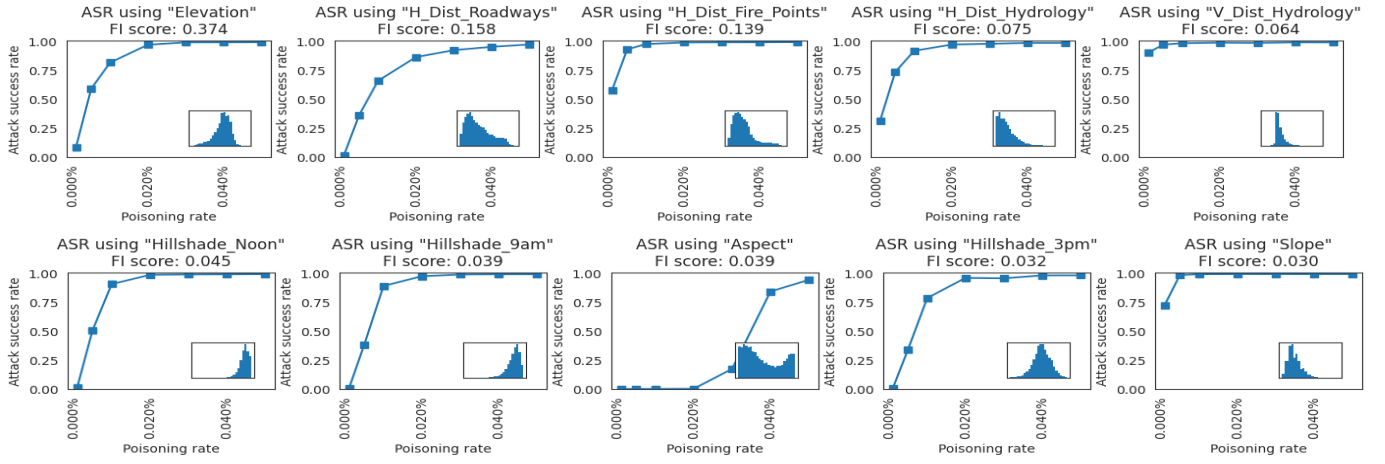


Fig. 15: ASR and feature distribution for FT-Transformer using features from top 5 and bottom 5 feature importance scores for the Forest Cover Type dataset.

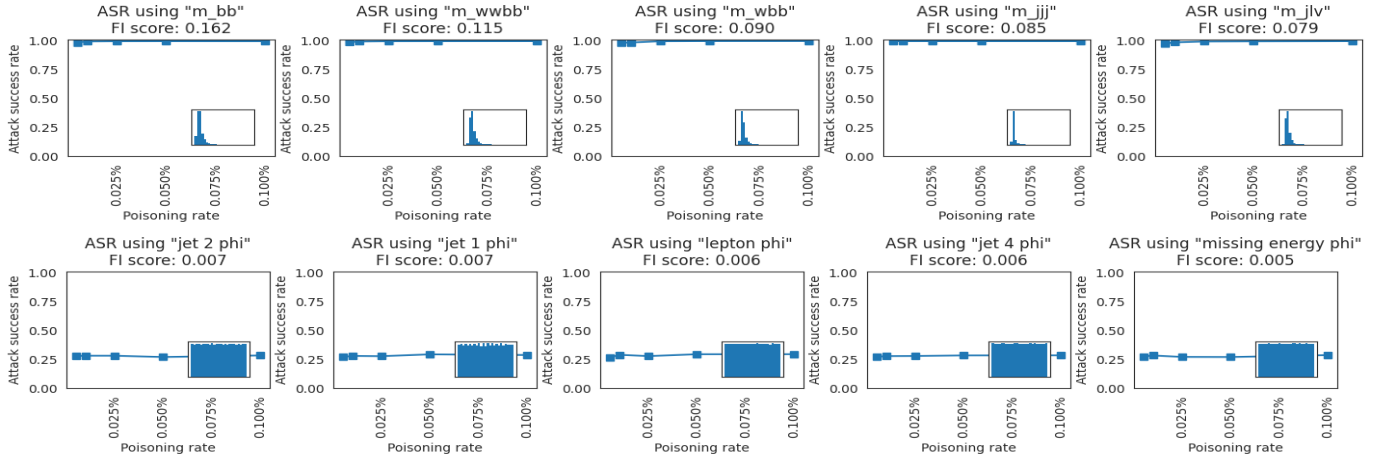


Fig. 16: ASR and feature distribution for FT-Transformer using features from top 5 and bottom 5 feature importance scores for the Higgs Boson dataset.

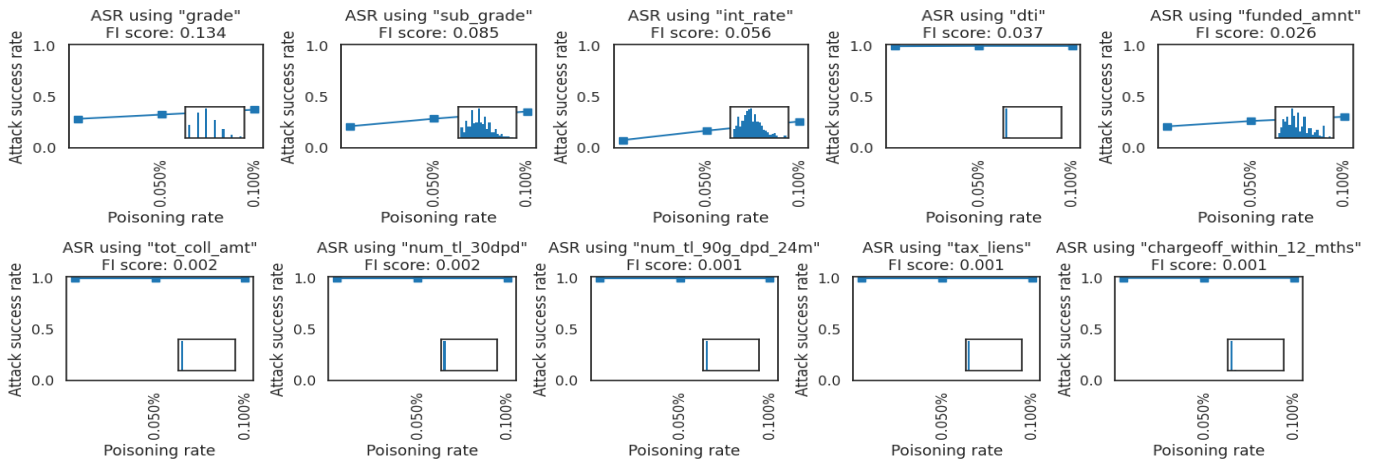


Fig. 17: ASR and feature distribution for FT-Transformer using features from top 5 and bottom 5 feature importance scores for the Lending Club dataset.

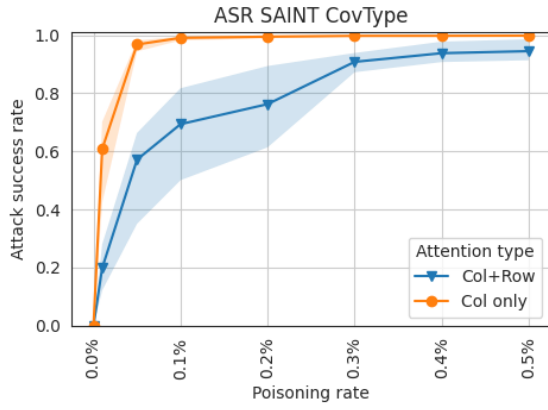


Fig. 18: ASR for SAINT on the CovType dataset (trigger size 1, out-of-bounds value) with and without row attention. Averaged over five runs.

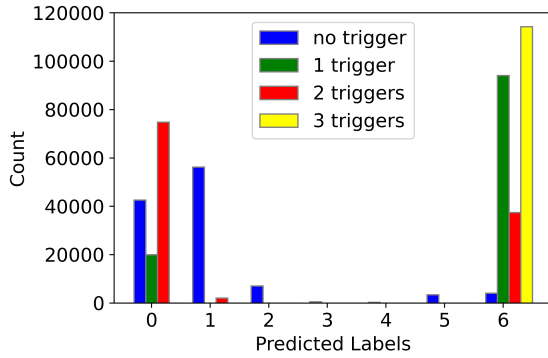


Fig. 19: Predicted labels distribution for different triggers on ASR test set (target label is 4) for the clean TabNet model on the Forest Cover Type dataset.

Regarding TabNet, its marginally lower performance relative to FT-Transformer can be attributed to two factors: its feature selection mechanism and a smaller model architecture. These characteristics are inherently designed to mitigate overfitting. As a consequence, TabNet might be less prone to learn a backdoor.

APPENDIX E

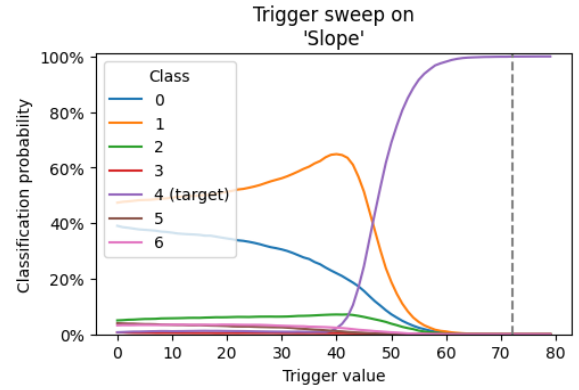
SUPPLEMENTARY INFORMATION FOR DEFENSES

A. Reverse Engineering-based Defenses

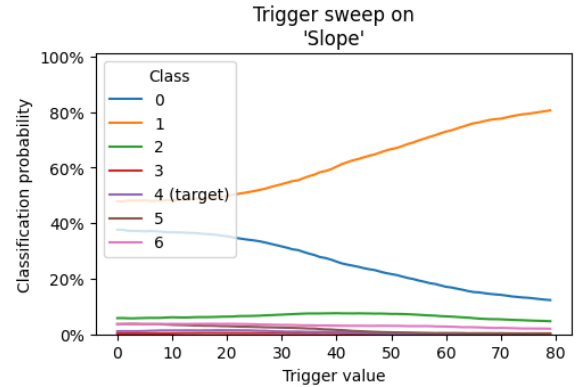
Figure 20, Figure 21, and Figure 22 show our sample results for how effective reverse engineering defense is in detecting triggers of sizes 1 and 2 for the CovType and LOAN datasets.

B. Spectral Signatures

Figure 23 and Figure 24 demonstrate correlation plots for the HIGGS and LOAN datasets, respectively. Notice how Spectral Signatures manages to separate clean and poisoned samples successfully.

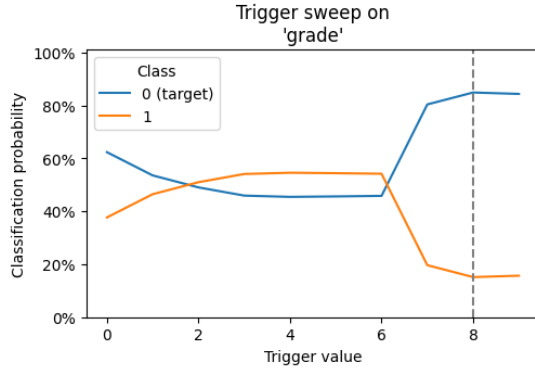


(a) Backdoored TabNet with slope trigger.

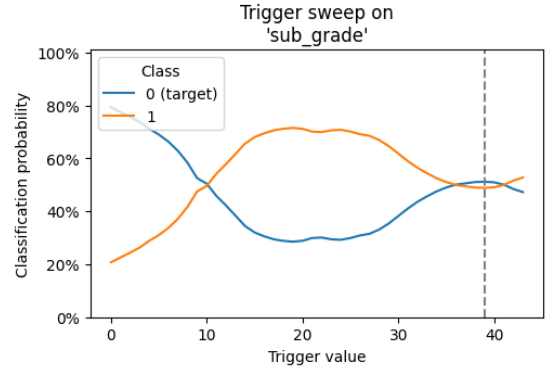


(b) Clean TabNet (slope).

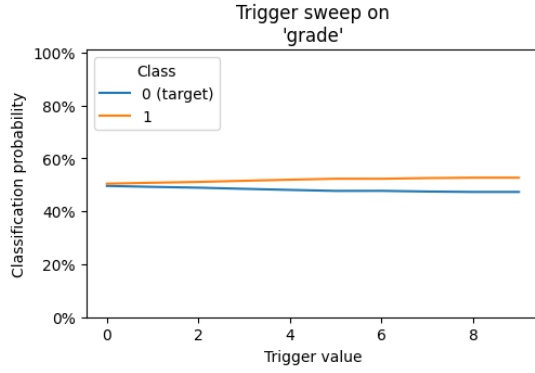
Fig. 20: Classification probabilities on the Forest Cover Type test set for different potential trigger values of the low importance feature `slope`. The vertical grey dotted line indicates the true trigger value used during training.



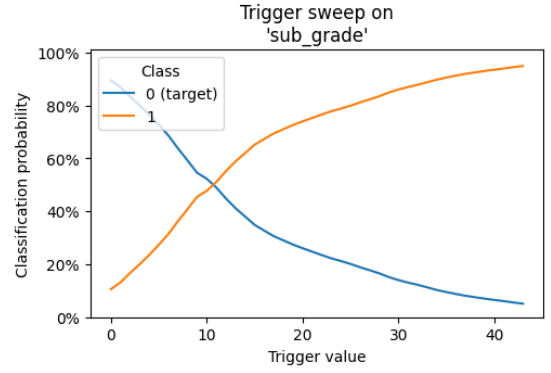
(a) Backdoored TabNet with size 2 (grade).



(a) Backdoored TabNet with size 2 (sub_grade).



(b) Clean TabNet (grade).



(b) Clean TabNet (sub_grade).

Fig. 21: Classification probabilities on the LOAN test set for different potential trigger values of `grade` in the trigger of size 2 consisting of `grade` and `sub_grade`. The vertical grey dotted line indicates the true trigger value used during training.

Fig. 22: Classification probabilities on the LOAN test set for different potential trigger values of `sub_grade` in the trigger of size 2 consisting of `grade` and `sub_grade`. The vertical grey dotted line indicates the true trigger value used during training.

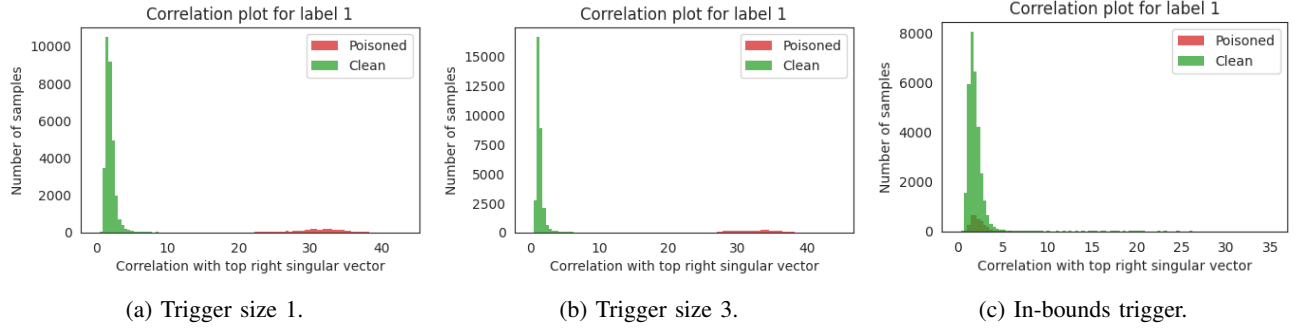


Fig. 23: Correlation plots for TabNet trained on the Higgs Boson dataset.

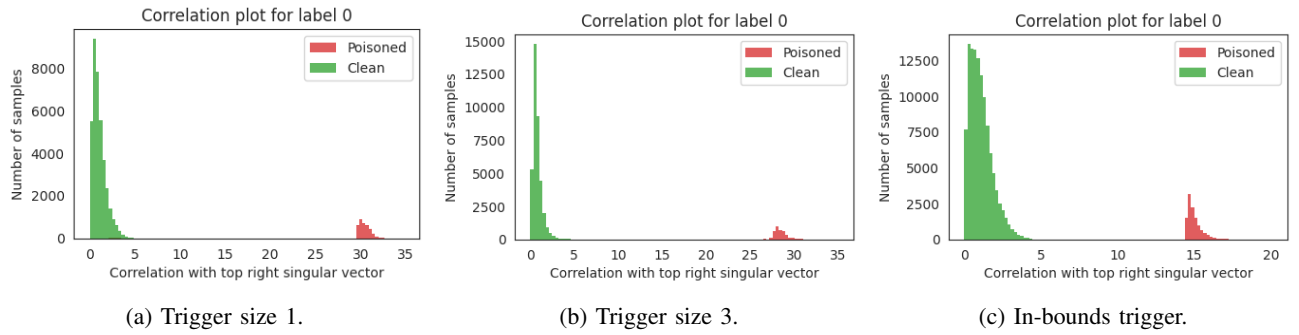


Fig. 24: Correlation plots for TabNet trained on the Lending Club dataset.