Master thesis
Computing Science

Radboud University

Backdoor Attacks against Transformer-based Neural Networks for Tabular Data

Author:
Bart Pleiter
S4752740

First supervisor/assessor:
Dr. Stjepan Picek
`stjepan.picek@ru.nl`

Second assessor:
Prof. M.A. Larson
`m.larson@cs.ru.nl`

July 31, 2023

Abstract

The training process of Deep Neural Networks (DNNs) usually requires large amounts of data and computing power, causing users to resort to untrusted data sources and the outsourcing of model training. As this reduces the control over the training process, it introduces the vulnerability to backdoor attacks. These backdoor attacks create hidden backdoors in the model, causing samples with a trigger to be classified as the class chosen by the attacker while maintaining the same behavior on benign samples. As of recently, transformer-based DNNs have shown to be competitive in the tabular data domain, which is currently dominated by gradient-boosted decision trees. Therefore, the threat of backdoor attacks now also applies to these new DNNs for tabular data.

This thesis investigates the threat of backdoor attacks against transformer-based DNNs for tabular data[1]. We explore different trigger generation methods and attack parameters. We found our tested models very susceptible to backdoor attacks, as we can achieve high attack success rates by only changing a single feature for a very low amount of samples. We also achieved high attack success rates with more stealthy variations of the attack, including a clean label attack. The trigger position was the most crucial parameter for a successful attack.

We also evaluated the performance and potential of several types of defenses against our backdoor attacks. We found most types of defenses not very successful against our attacks. Only the Spectral Signatures defense, which separates poisoned samples from clean samples in the training set based on statistics in the latent space, was able to detect and remove the backdoor.

As our backdoor attacks are both effective and hard to defend against, we hope to see future research further investigate these types of attacks and how to defend against them in the tabular data domain.

[1]Our code can be found at `https://github.com/bartpleiter/tabular-backdoors`

Contents

Chapter 1

Introduction

Deep neural networks (DNNs) have become a popular and well-performing solution for various tasks within the image, text, and audio domains. Since DNNs usually require lots of data and computing power to train, users may choose to outsource the training to third parties or use pre-trained models. Furthermore, the large amounts of data required to train these models might make users use untrusted data from the internet or via crowdsourcing. This, in turn, limits the amount of control the user has over the model's training process, introducing potential security risks. A popular type of attack that exploits this lack of control is the backdoor attack. Backdoor attacks embed hidden backdoors into the model, usually via data poisoning, to change the predictions during inference on samples containing a trigger without changing the predictions on benign samples. Backdoor attacks have been studied extensively since they were introduced in 2017[19]. However, this research was mainly done in the image domain.

DNNs are usually applied on tasks with homogeneous data like images or text since they perform well on this type of data. How to successfully apply DNNs on heterogeneous data like tabular data, where usually classical machine learning solutions like gradient-boosted decision trees perform best, is still an active research area [5][41][18]. Recently, DNNs specifically made for tabular data are being developed, increasing the performance of DNNs on tabular data. One of the main recent approaches is the use of transformer-based models.

There are many scenarios with machine learning on tabular data where an adversary has the incentive to influence the results of the machine learning model. An example would be in a financial setting where the adversary wants to apply for a loan. The adversary would have a clear incentive to cause somehow the model to predict that the adversary can pay back this loan, while under normal circumstances, it would predict the opposite. For this reason, it is important to look at backdoor attacks in a tabular data setting.

Given the importance of backdoor attacks in the tabular data domain, in this thesis, we explore backdoor attacks for DNNs on tabular data in various settings. We attempt to answer the following research question and sub-questions:

- How vulnerable are transformer-based neural networks for tabular data against backdoor attacks?

    1. How can we embed a hidden trigger in tabular data, given the challenges of tabular data compared to images and text?
    2. What are the most relevant parameters for a backdoor attack on tabular data?
    3. How do different trigger generation methods affect the performance of the backdoor attacks?
    4. How can we defend against backdoor attacks on tabular data?

We summarize the main contributions of this thesis as follows:

1. We find transformer-based DNNs for tabular data to be vulnerable to backdoor attacks. By changing a single feature to a value slightly higher than the maximum recorded in the training data, we achieved high attack success rates with low poisoning rates on all models and datasets.

2. We adapt a backdoor attack designed for federated learning to a centralized setup, creating what we believe to be the first backdoor attack that can be applied to most tabular datasets for classification tasks.

3. We show two more stealthy attack variations, an attack using in-bounds trigger values and a clean-label attack, that perform well against our tested models.

4. We find the trigger position to be the most important parameter to the backdoor attack. While we find that features with high feature importance scores generally lead to higher attack success rates, we also observe there are more currently unknown parameters that make a feature suited for a backdoor attack.

5. We explore several defenses against our backdoor attacks. We find that Spectral Signatures can successfully defend against our attacks in most cases.

This thesis is structured as follows. In Chapter 2, we provide the terminology and relevant background on both backdoor attacks and neural networks for tabular data. A review of the related work for this thesis is presented in Chapter 3. Then, in Chapter 4, we explain the challenges of generating a backdoor trigger in tabular data compared to other domains and discuss our methodology for this research. An overview of the data used in this research is presented in Chapter 5. In Chapter 5, we present our experimental setup and the results of our experiments. Our findings from these results are discussed in Chapter 7, where we also provide directions for future research. Finally, we conclude our research in Chapter 8.

Chapter 2

Background

This chapter provides relevant background information to help the understanding of this thesis.

2.1 Terminology

This section aims to give a clear overview of several commonly used technical terms in research on backdoor attacks, including ours.

- **Clean model** - is a machine learning model without malicious or hidden functionalities.
- **Backdoored model** - is a machine learning model containing one or more hidden backdoors.
- **Trigger** - is a value or pattern that activates a backdoor in a backdoored model.
- **Target label** - is the attacker-specified label that the attacker wants all samples containing the backdoor trigger to be predicted as.
- **Poisoned sample** - is a training sample used to create a backdoor in a machine learning model.
- **Attack Success Rate (ASR)** - is the accuracy of a machine learning model on poisoned samples towards the target label.
- **Clean Data Accuracy (CDA) or Benign Accuracy (BA)** - is the accuracy of a machine learning model on clean samples.
- **User** - is a non-malicious person who wants to train a machine learning model without a backdoor.

2.2 Backdoor attacks

A backdoor attack attempts to create a hidden backdoor in a machine learning model, causing samples containing a trigger to be classified as the target label while not changing the prediction on clean samples. One of the first backdoor attacks was BadNets[19], which allows the creation of backdoors in image classifiers. It became the classic example of a backdoor attack, as it has been cited over a thousand times. BadNets can be described as the basic form of a backdoor attack. It works by adding a trigger, which can be either a pixel pattern or sticker, to some of the input images used for training the image classifier. After adding the trigger, the label is changed to the target label.

2.2.1 Differences compared to poisoning and evasion attacks

While backdoor, poisoning, and evasion attacks appear quite similar as they all attempt to alter the outcome of a machine learning model, it is good to understand the different methods they use to achieve this goal.

Poisoning attacks Poisoning attacks influence certain stages of the training process where the attackers attempt to degrade either the performance of specific examples (targeted) or of the entire model (untargeted)[44]. Data poisoning attacks modify the training data[44], code poisoning attacks modify the implementation of the training algorithm like the loss function[50], and model poisoning attacks modify the model itself like its weights[55].

Backdoor attacks Backdoor attacks can be seen as a form of poisoning attacks, where the attacker still attempts to misclassify inputs, but only when a trigger is present. These attacks are more stealthy than poisoning attacks, as the performance of the backdoored model on the clean data is ideally the same, making it more difficult for the user to detect the presence of a backdoor.

Evasion attacks Evasion attacks [34] differ from backdoor and poisoning attacks as they are only performed during inference time. In this case, the attacker attempts to craft a specific input that is misclassified, either targeted or untargeted, by the model. They do not require modifications to the model or training data. While black box evasion attacks do not require knowledge about the model, grey and white box attacks do require knowledge about the model, for example, to obtain the gradient to move towards a different class.

2.2.2 Main variations on backdoor attacks

After the first backdoor attacks were introduced, many attack variations were proposed. This section describes some of the main categories of backdoor attacks.

Clean-label backdoor attacks

The BadNets attack is an example of a dirty-label attack, as the label of a poisoned sample is changed to the target class. This can make it obvious that the training data has been tampered with, as the poisoned samples are inconsistent with their labels. Clean-label attacks[46][53] differ from dirty-label attacks in that they do not change the label of poisoned samples to the target label. This makes them generally more stealthy and difficult to detect. In the image domain, it has been shown that by just restricting the dirty label attack like BadNets[19] to only poison samples with the target label, the attack does not work anymore, even with very high poisoning rates [46]. This is because the model still learns from the features of the sample[15]. Therefore, clean-label attacks often use strategies to force the model to learn from the trigger by making the original input harder to classify. For example, this can be done using a Generative Adversarial Network (GAN) or adversarial examples during training time[46].

Dynamic backdoor attacks

Another variation of the backdoor attacks is the dynamic backdoor attack. The idea of dynamic backdoor attacks is that the trigger is not fixed for all inputs like with a static attack such as BadNets. The advantage of a dynamic attack is its ability to bypass certain defenses that use the staticness of the trigger[39], like trigger reverse-engineering defenses.

Most dynamic backdoor attacks are sample specific[32][28][52], which means that the generated triggers depend on the clean sample it is applied on. These triggers are usually generated using a GAN or autoencoder trained together with the backdoored model. An example of such an attack is done by Nguyen et al.[32] using an encoder-decoder network that takes an image as input and outputs a trigger pattern specific to that image. To make this attack work, two conditions need to be enforced. First, every input should have its unique trigger. This is enforced by including a diversity loss during training, which is high when the trigger for a random other input is similar to the current one. Second, the trigger for one input should not be usable for a different input. This is enforced using a cross-trigger loss, which is the standard training loss (usually cross entropy for classification) for a trigger applied on a different input with a clean label. For this type of attack to work, the attacker needs full control over the training procedure.

Invisible backdoor attacks

For backdoor attacks like BadNets, a human inspecting the dataset could clearly spot the trigger in a sample. Invisible backdoor attacks try to prevent the detection of a trigger by human inspection. This means that a poisoned sample with an invisible trigger should not be distinguishable by a human from a clean sample. In combination with a clean label attack, invisible backdoor attacks can easily fool humans inspecting the data. In the image domain, invisible backdoor attacks have been performed using techniques like steganography[26] (e.g., by embedding the trigger in the least significant bits of each pixel value) and optimization of generated trigger patterns similar to adversarial examples[26][54]. In the text domain, this can be performed using style transfer[36]. In the audio domain, this can be performed using a trigger outside of the range a human can hear[24].

2.3 Neural networks for tabular data

DNNs have become state-of-the-art in the image, text, and audio domains, as they work well with large amounts of homogeneous data. Tabular data, one of the most commonly used data types in machine learning, is one of the few remaining domains where DNNs have yet to outperform other machine learning methods. As of writing, classical machine learning algorithms like gradient-boosted decision trees are the most popular choice for tabular data, as they perform well and relatively fast, even on smaller amounts of data. Recently, several DNN architectures were developed specifically for tabular data in an attempt to improve the performance of DNNs on tabular data, showing competitive performance[35][2][42][16][21][5]. These specialized architectures generally fall within two categories: Hybrid models and transformer-based models. Hybrid models like NODE[35] combine classical machine learning algorithms with neural networks, while transformer-based models attempt to adapt the successful transformer architecture to the tabular domain.

Transformer-based models are inspired by transformer networks[47], which are currently very successful in the image domain (e.g., vision transformers[11]) and the text domain (e.g., BERT[8] and GPT-3[6]). Transformer models were initially introduced for language tasks like language modeling and machine translation. They use self-attention to weigh the importance of different words in a sequence when computing the representation of a sequence. Furthermore, they allow for parallel processing of the input tokens, as each word in the sequence is positionally encoded. An advantage of self-attention is that it allows for some form of explainability, as it indicates which parts of the input were important for predicting the output. One of the first and most well-known models based on the transformer architecture is TabNet[2].

Chapter 3

Related Work

3.1 Backdoor attacks

To the best of our knowledge, the first backdoor attack was BadNets[19], which introduced the backdoor attack in the image domain. The authors showed how to create a backdoor in a Convolutional Neural Network for the MNIST[25] dataset by adding a single-pixel or pattern-based trigger to some of the training samples and changing the label to the target label. This resulted in an almost perfect attack success rate without a significant drop in clean data accuracy. They then applied this attack to a traffic sign classifier using different small images (e.g., a yellow sticker) as a trigger. Again, the attack was successful and was shown to be working in the physical domain as well. After BadNets, various research was done on variations of this attack using different methods, domains (e.g., text data), and trigger types. This research is well summarized in a recent survey paper by Li et al.[27].

3.1.1 Clean-label attack

In work by Turner et al.[46], the authors introduce the clean-label attack where the attacker only poisons samples of the target label, causing all poisoned samples to be label-consistent. The authors show that by limiting the data poisoning performed in the BadNets attack to samples from the target label, the attack fails. They argue the attack fails because the model is not forced to learn a backdoor, as there is enough information in the poisoned sample without relying on the trigger. They then propose two methods using either a GAN or adversarial examples to overcome this issue by making the poisoned samples harder to classify while keeping them visually similar to humans.

3.1.2 Evaluation of backdoor attacks

Abad et al.[1] analyze relevant backdoor attack parameters for the image domain. By investigating backdoors in several models and datasets, the authors give more insight into the effect of parameters such as model architecture, trigger size, trigger position, trigger color, and poisoning rate.

While, as far as we know, the stealthiness of existing backdoor attacks on tabular data is not discussed, several works on evasion attacks on tabular data[17][3][31] do discuss the perceivability of perturbations to tabular data. However, there is no clear consensus on measuring this and therefore remains an open question.

In the text domain, Lyu et al.[30] provided more insights into the effect of backdoors in transformer models. They showed how, in a backdoored transformer, the attention of a number of heads in the multi-head attention mechanism drifts toward the trigger token. The authors then propose a defense based on this observation by generating trigger candidates and evaluating the attention-drifting behavior on the test model.

3.1.3 Backdoor attacks on models for tabular data

Only few works consider the possibility of backdoor attacks on models for tabular data. Two works by Joe et al.[22][23] show backdoor attacks are effective against models for mortality prediction using Electronic Health Record (EHR) data in tabular form. Each EHR sample consists of 17 features measured over 48 hours with one-hour intervals, making them time-series data. The authors propose two different trigger-generation methods to create triggers resulting in poisoned samples with the same characteristics as clean data, making them difficult to detect.

The first method the authors describe[22] uses the temporal covariance of the 17 features to generate an additive trigger. The trigger strength is calculated using the Mahalanobis distance. The reason the authors use temporal covariance between the features is to generate realistic values, as there are dependencies between features over time. The example the authors give for this is the height and blood pressure features. Height is not expected to change during 48 hours, while blood pressure can be expected to change over 48 hours. Therefore, a trigger should perturb the blood pressure feature more than the height feature over 48 hours. Using their trigger generation method, the authors achieve an ASR of 97 percent using a less than five percent poisoning rate. While these are promising results, the authors only perform their attack on Logistic Regression, Long-Short Term Memory, and Multi-Layer Perceptron (MLP) models without giving any details on the implementation, training process, or code availability. Furthermore, how to generalize this backdoor attack to generic tabular data remains a challenge, as their trigger generation method makes use of the time-series structure of EHR data.

The second method the authors describe[23] makes use of a Variational Autoencoder to create a missing value mask consistent with the missing value pattern of the data. This mask is then used as a trigger. In addition to the models used in the first method, the authors also perform this attack on a Gated Recurrent Unit. Again, no details or code were presented for this research. Additionally, this trigger generation method requires the missing value mask to be used as input for the machine learning model, making it unsuited for generic tabular data.

Xie et al.[51] presented a distributed backdoor strategy in a Federated Learning setting. By splitting the trigger into four segments over four malicious clients, the authors were able to bypass aggregation rules, causing a backdoor in the model. This backdoor could then be triggered using the combined segments. The authors performed their backdoor attack on four datasets, including a tabular dataset for loan status classification. For the tabular dataset, a trigger of 8 features was used (two per malicious client), each feature having a static trigger value slightly higher than the maximum recorded value in the dataset. A relatively high poisoning rate of 15.625 percent was used to perform this attack on a relatively simple MLP model with three fully-connected layers. The authors also investigated certain factors affecting the performance of their backdoor attack, including the trigger location. For the tabular dataset, the authors found using low-importance features achieved higher attack success rates than using high-importance features.

3.2 Defenses against backdoor attacks

Several defenses have been proposed over the years, mainly for the image domain.

Neural Cleanse[48] attempts to reverse-engineer the trigger using the potentially backdoored model and a set of clean samples. If a potential trigger crosses some threshold based on an outlier-detection algorithm, the backdoor for that trigger is removed using either a neuron-pruning or unlearning strategy.

Spectral Signatures[45] attempts to filter poisoned samples from the training dataset after the model is trained. It does so by performing statistics on the input representations in the latent space of the potentially poisoned model. For this, it assumes poisoned and clean samples are separable in latent space. Other defenses based on the same assumption are Activation Clustering[7], SCAn[43], and SPECTRE[20]. After identifying the poisoned samples, the model is retrained from scratch without the detected poisoned samples.

Februus[9] uses GradCAM[40] to identify regions of the input image during test time that contribute the largest to the output prediction, with the idea that this region likely contains the trigger in case of a poisoned sample. After removal, a GAN is used to restore the removed areas.

STRIP[14] attempts to detect and filter poisoned samples during test time. It works by superimposing other inputs on top of the sample. By observing the randomness in the prediction probabilities, a decision can be made whether the sample contained a backdoor. If the randomness is low, the model could confidently predict most of the superimposed samples, indicating the presence of a trigger.

Chapter 4

Methods

In this chapter, we explain our methodology for answering our research questions. We first describe the threat model, followed by the model and dataset selection for our experiments. Then, we describe our methodology for performing our backdoor attacks, followed by backdoor defenses.

4.1 Challenges of tabular data compared to other domains

Tabular data differs from image or text data in various ways, making it non-trivial to adapt a trigger from the commonly used image or text domain to the tabular data domain. This section will discuss the challenges of creating a backdoor trigger for tabular data.

Heterogeneous data Tabular data are usually heterogeneous, while image, text, and audio data are homogeneous. This means that in tabular data, each feature can be of a different type with different statistics and distributions. Therefore, we cannot always apply the same backdoor trigger value for each feature. An example would be an income dataset with the features *name*, *age*, and *income*. The *name* feature is a categorical feature containing strings of characters, the *age* feature is a numerical feature containing only integers between 0 and 100, and the *income* feature is also a numerical feature but contains numbers from a very different range and distribution than the *age* feature. For a potential backdoor trigger on this dataset, a value of 40,000 in the *income* feature could make sense but will make no sense at all for the *age* or *name* feature.

With homogeneous data, all features have the same range and similar statistics. For example, the commonly used MNIST dataset[25] contains monochrome images of 28 by 28 pixels. Here, each pixel of a sample image can be seen as a feature with a value between 0 and 255. For a potential backdoor trigger, any value between 0 and 255 would make as much sense on each of the 28 times 28 features.

Furthermore, certain feature values are mutually exclusive. An example of this is a one-hot variable. A potential backdoor trigger that sets multiple of these features to 1 results in a sample that does not make sense and, therefore, could be more easily detected.

No spatial dependencies between features In contrast to images, text, and audio, tabular data usually have no spatial dependencies between features. Shuffling the order of the features in a tabular dataset will likely have no effect on the performance a machine learning model can achieve while shuffling the pixel positions in an image dataset will. Therefore, we cannot use the spatial dependencies between features for a model to learn a backdoor trigger similar to a pixel pattern in the image domain[19][4].

Single feature importance While perturbing a single pixel in an image will likely not change the output prediction for most image classifiers, perturbing a single feature on tabular data can significantly change the output prediction, depending on the feature's importance. This suggests that feature selection for the backdoor trigger might influence the performance of the backdoor attack. Furthermore, this can present a challenge for reverse-engineering defenses, as shown in section 5.6.1.

4.2 Threat model

For our backdoor attacks, we define the following threat model.

Attacker's goal The attacker's goal is to insert a hidden backdoor into the machine-learning model of the user. The attacker does not want the user to know or find out about the presence of the backdoor. Ideally, during inference time, this backdoor causes samples containing a trigger to be classified into the class chosen by the attacker. Samples without such a trigger should be classified similarly as a clean model. Furthermore, the attacker does not want the trigger to occur naturally in the data, as this causes unwanted activations of the backdoor, which can raise suspicion by the user.

Attacker's knowledge For our threat model, we assume a grey-box setting. In this setting, the attacker has no knowledge of any training parameters, such as the model used and the training parameters. However, the attacker has knowledge of at least some parts of the training data, allowing the attacker to get a general idea of each feature's distribution. This allows the attacker to know for each feature their general range and common values, as this is needed to select trigger values for the attacks we present in this thesis.

Attacker's capabilities To achieve their goal, the attacker can only modify or poison a small number of samples (in the order of a few percent) during the training process. The attacker cannot modify any other part of the training process, such as the model's structure, weights, or parameters. During inference time, the attacker can query the model with any possible input sample the model allows.

Scenarios Our described threat model applies to two realistic scenarios; the outsourcing training of the model to a third party and the usage of compromised training data. In the outsourced training scenario, the user outsources the training of the model to a malicious or compromised third party. The main reason for choosing to outsource the training process could be the expensive training of DNNs in terms of time and hardware. In this scenario, the user verifies the performance of the trained model using a held-out test set containing clean samples. In the compromised training data scenario, the adversary has gained access to (part of) the dataset in some way. There are many reasons why this could happen. For example, the user might have downloaded (part of) the dataset from an untrusted source over the internet, or the user used crowd-sourced data. We do not consider the scenario of malicious pre-trained models, as these are less relevant on tabular data since, generally, knowledge from one tabular dataset does not transfer well to other tabular datasets.

4.3 Selection of transformer-based DNNs for tabular data

For our experiments, we selected three different state-of-the-art transformer-based DNNs for tabular data to show that our results are not limited to a specific model architecture.

We selected TabNet[2], FT-Transformer[16] and SAINT[42]. We chose TabNet because of its popularity and decision tree-inspired sequential architecture for tabular data. This specialized architecture makes it quite different from the other transformer-based models. We chose FT-Transformer because of its good performance and mostly unmodified architecture from the original transformer model[47]. Finally, we chose SAINT because of its intersample attention mechanism and good performance.

4.3.1 TabNet

TabNet[2] is one of the first transformer-based DNNs specifically made for tabular data. Since decision tree-based models generally perform best on tabular data, the authors attempted to create a novel DNN architecture, completely trained using gradient decent-based algorithms, with the strengths of decision trees. Decision trees are able to select features with the most information gain. According to the authors, TabNet can also do this using instance-wise feature selection using attention mechanisms like a transformer network. Furthermore, TabNet uses a sequential architecture like decision trees in which each decision step contributes to the final decision. It uses non-linear processing of the selected features, as is common with neural networks. Using the feature selection masks at each decision step, TabNet also allows for some form of interpretability at each decision step and globally.

4.3.2 FT-Transformer

FT-Transformer[16] is a simple adaptation of a transformer network for tabular data. It shows very competitive results compared to other DNN-based models for tabular data. FT-Transformer stands for Feature Tokenizer + Transformer. The feature tokenizer first creates embeddings using the input features. Then, a stack of transformer layers is applied to these embeddings. A [CLS] token is added to the end of the input for classification, similar to transformer networks from other domains. In contrast to transformer networks from other domains, no positional encoding is needed as the position of the features in tabular data is not relevant for classification.

4.3.3 SAINT

SAINT[42] is in most aspects similar to FT-Transformer as it also creates embeddings of the input features, after which a stack of transformer layers is applied. The main difference is that each transformer layer consists of a novel intersample attention block on top of the standard transformer block. This intersample attention block performs attention between rows and allows the model to "borrow" missing or noisy features from similar samples in the same batch. The authors show that the addition of intersample attention improves performance.

4.4 Data

For this research, we selected a diverse range of datasets to ensure the generalizability of our experiments rather than being limited to specific datasets. Table 4.1 shows an overview of the datasets we used. We used the following criteria to select our datasets.

First, we chose datasets with a classification task (binary or multi-class) rather than regression since most of the existing research on backdoor attacks is on classification tasks from other domains. We argue this might make it easier for us to translate parts of the attacks and defenses from other domains to the tabular data domain.

	Forest Cover Type	Higgs Boson	Lending Club LOAN	SYN10
Samples	581,012	11,000,000	588,892	100,000
Numerical features	10	28	60	10
Categorical features	44	0	8	0
Target labels	7	2	2	2

Table 4.1: Overview of used datasets in this research. Values are calculated after preprocessing

Secondly, we chose datasets with a large number of samples, as it has been shown that DNNs generally perform better with larger datasets than smaller datasets[18], which is also a more realistic scenario for deep learning. We chose datasets with more than 100,000 samples.

Finally, to give some headroom for the trigger generation, we want to have enough features available to include in our triggers without altering the sample too much. Especially numerical features are important as they usually allow for many different values. Therefore, we chose datasets with at least ten numerical features.

4.4.1 Popular datasets

To show that our experiments are not limited to very specific datasets, we included two commonly used datasets in research on DNNs for tabular data: Forest Cover Type and Higgs Boson[12].

Forest Cover Type (CovType)

The Forest Cover Type dataset[1] contains cartographic variables for each 30 by 30-meter area and the corresponding forest type. It is used in several papers related to DNNs on tabular data[18][41][2][42][16]. The target label contains one of seven possible forest types, with some types being more present in the dataset than others as this dataset is not well balanced. The 44 categorical features represent two one-hot encoded variables with 4 and 40 possible categories. The task for this dataset is relatively easy as the models we tested were able to obtain an accuracy of around 95 percent. We did not perform any significant preprocessing on this dataset.

Higgs Boson (HIGGS)

The Higgs Boson dataset[2] is used to classify between particle collision events that produce Higgs boson particles and those that do not produce these particles. It is a relatively large dataset with 11 million samples and is well balanced, with a ratio of 53 positive to 47 negative samples. The dataset has 28 features, of which 21 are measurements from the particle detectors, and 7 are derivations of these 21 measurements. This dataset is used in several papers related to DNNs on tabular data[18][41][2][16] and is a more challenging dataset as the models we tested were able to obtain an accuracy of around 75 percent.

[1]`http://archive.ics.uci.edu/ml/datasets/covertype`
[2]`https://archive.ics.uci.edu/ml/datasets/HIGGS`

4.4.2 Realistic dataset

While the two previously mentioned datasets are commonly used in research on DNNs for tabular data, the tasks they are used in are not particularly susceptible to backdoor attacks. To show that backdoor attacks on tabular data could be a real threat, we chose a financial dataset and task which is more likely to be targeted by an attacker.

Lending Club Loan dataset (LOAN)

LendingClub used to be the larger peer-to-peer lending platform where investors could invest in loans from other people. The general idea was that borrowers with higher credit scores would have to pay less interest compared to borrowers with lower credit scores. Throughout many years, LendingClub has provided data on accepted and rejected loans on their platform, including their corresponding loan status indicating whether the loan was fully paid back. For an investor, it would be useful to use this data to create a model that predicts whether a loan will be paid back or not, given the information provided by the platform. A possible attacker then could be someone with a low credit score who wants a loan. Creating a backdoor in the model might fool the investor into thinking their loan will likely be paid back. This, together with the high-interest rate because of the low credit score, the attacker will be much more likely to get a loan compared to other persons on the platform.

The dataset we used was obtained from Kaggle[3]. We used the accepted loans since they contain the information needed to classify the loan status. As this dataset also contains features with information about the loan after it has been issued, we removed all features which were not available to investors before the loan was issued[4].

As for preprocessing the dataset, we did the following. We removed all features that are not visible to investors, have more than 30% missing values, or are irrelevant to the task, like "url" and "id". Furthermore, we separated features with a date into separate year and month features and label-encoded all categorical features. We also removed the zip code feature since the TabNet implementation we used has an issue where it cannot handle categorical features with a very high number of dimensions. Finally, we categorized the target label "loan_status" into good and bad investments while dropping ongoing loans. Good investments contain the loans that were paid off in time, while bad investments contain loans with a loan status of "late", "default", or "charged off". All remaining missing values were inputted with zero. The resulting dataset is quite imbalanced, with a ratio of 78.5 good investments to 21.5 bad investments. Since we have plenty of samples, we decided to perform random undersampling to balance the dataset and reduce the runtime of the experiments. This results in a balanced dataset of about the same size as the Forest Cover Type dataset. The models we tested obtained an accuracy of around 67 percent on this dataset, as it is a very difficult task to predict the loan status using only the provided features.

4.4.3 Synthetic dataset for feature importances (SYN10)

To further investigate the relation between feature importance and attack success rate when using a single feature as the backdoor trigger (see section 4.5.1), we created a synthetic dataset using the make_classification method from the scikit-learn Python library[33]. Using this method, we created a dataset for binary classification with five informative features drawn from two Gaussian clusters per class placed around the vertices of a five-dimensional

[3] `https://www.kaggle.com/datasets/wordsforthewise/lending-club`

[4] The list of features available to investors before the loan was issued was obtained from the "browseNotes" table from `https://www.kaggle.com/datasets/adarshsng/lending-club-loan-data-csv?select=LCDataDictionary.xlsx`

hypercube, and five non-informative features drawn from Gaussian noise. The resulting dataset is balanced and has 100,000 samples.

4.5 Trigger generation methods

As described in section 3.1.3, backdoor triggers for tabular data have not been well explored. As a starting point, we adapt the trigger generation method by Xie et al.[51], which was used in a federated learning setting, to our centralized setup. Their trigger generation method can be described as follows. First, six low-importance features are selected as the trigger, each with a trigger value slightly larger than their maximum values (which we will refer to as an out-of-bounds trigger). Then, the six trigger features are evenly distributed over three malicious clients, resulting in two trigger features per client. Finally, the target label is flipped to the target label for each poisoned sample.

Since we use a centralized setup, we will not split the trigger into smaller parts over multiple clients but instead use the full trigger when poisoning the data. For this trigger generation method, we explore the effect of using different trigger locations (feature selection) based on feature importance. We also explore the effect of using different trigger sizes (number of features). Then, to increase the stealthiness of our attack, we extend this method by using in-bounds trigger values (feature values), followed by performing a clean label attack where we do not change the labels of the poisoned samples.

4.5.1 Trigger location

Xie et al. show that the selection of the trigger location can significantly impact the ASR[51]. More specifically, they show that using features with low feature importance scores obtained by several decision trees results in a higher ASR than using features with high feature importance scores. These results align with their experiment on the trigger location for image data, where they show that moving the trigger toward the center of the image, containing the important pixels for the main task, causes the ASR to drop. Other experiments on backdoor attacks in the image domain by Abad et al.[1] show that there is no universal best trigger position and that the effectiveness depends on the dataset's properties and how the model learns.

We will perform the following experiment to investigate the trigger location importance in our setting. We first calculate the feature importance scores and rankings similarly to Xie et al., followed by measuring the ASR and BA for different poisoning rates for each numerical feature, dataset, and model combination. This allows us to observe the effect of feature importance on the attack performance. To further investigate the relation between feature importance and attack success rate, we included our synthetic dataset (see section 4.4.3) in these experiments.

Feature importance scores and ranking

To calculate feature importance scores and rankings, we use the feature importances from several popular decision tree classifiers in addition to the global feature importance scores available from TabNet[2]. For the decision tree classifiers, we chose XGBoost, LightGBM, CatBoost, and Random Forest because of their popularity. We train each model three times on each dataset. We then average the feature importance scores for all numerical features over all three runs and scale the absolute values to sum to one.

Feature importance vs. attack performance

To investigate how the feature importance relates to backdoor attack performance, we train backdoored models using a single feature with a trigger value of 10 percent outside of the range of the feature, which means a value v of $max(v) + (max(v) - min(v)) * 0.1$. We repeated this for each feature, model, and dataset combination for different poisoning rates. To decrease the runtime of this experiment, we reduced the HIGGS dataset to 500,000 samples using random sampling without replacement. Using the average feature importance we obtained before, we can see how the ASR and BA change as the feature importance score changes. We average our ASR and BA scores over three runs.

4.5.2 Trigger size

Xie et al.[51] included multiple features in their trigger for their backdoor attack. As shown in the image domain, a larger trigger size can significantly increase the attack success rate[1]. However, using a larger trigger size also makes the trigger easier to detect as more perturbations to the original sample are made. We hypothesize this also holds in our setting with tabular data. To verify this and to see how much the trigger size affects the backdoor performance, we train backdoored models using either one, two, or three features, again each with a trigger value of 10 percent outside of the range of the feature. For our trigger positions, we selected the top three important features according to section 4.5.1 as we found (see section 5.2) that, in general, the most important features lead to higher attack success rates. See appendix A for the values we used.

4.5.3 In-bounds trigger values

Our current method of choosing a trigger value consists of setting the value slightly higher than the maximum recorded value, which we call an out-of-bounds value. An advantage of using such a trigger value is that it does not occur in the training data. Therefore, it lowers the odds of false positives that could negatively affect the clean data accuracy. Furthermore, it might also be easier to learn for a model, as this value is unrelated to other labels than the target label.

A disadvantage of using an out-of-bounds value is that it might be easier to detect by the user. The trigger might be detected as outliers or extreme values if the user has access to the training data (as in the compromised data scenario). Even without access to the training data, very large values might also be suspicious during inference time, making the trigger less stealthy. Furthermore, an out-of-bounds value might not be allowed for some features, as there might be a maximum value, as discussed in section 4.1. This is mainly the case for categorical features.

To overcome these disadvantages and to make our attack more stealthy, we will perform a backdoor attack similar to section 4.5.2, where instead, we use the most common value of each trigger feature as the trigger value (which we will refer to as an in-bounds trigger). As these trigger values are the most common in the training data, we fix the trigger size to three to make the combination of each trigger value less common. In fact, with a trigger size of three, we did not find any sample in our datasets that matched the trigger. See appendix A.2 for the values we used.

4.5.4 Clean label attack

To further increase the stealthiness of the backdoor attack, we also perform a clean label attack where we only poison samples from the target class. This makes poisoned samples in the training data harder to detect as the samples are now consistent with their label.

This increased stealthiness usually comes with the cost of requiring more poisoning or other methods to force the model to learn the trigger since the model can learn from the other features instead as they now are label consistent. It has been shown in the image domain that by just restricting a dirty label attack like BadNets[19] to only poison samples with the target label, the attack does not work anymore, even with very high poisoning rates [46]. We test whether this also holds for our setting with tabular data. For this experiment, we perform a backdoor attack similar to section 4.5.2 with a trigger size of one while only poisoning samples of the target class.

4.6 Evaluation metrics

To evaluate the performance of our backdoor attacks, we use the following two commonly used metrics:

Attack Success Rate (ASR) The Attack Success Rate indicates how well the backdoored model can predict test samples with a trigger as the attacker-targeted class[13]. We obtain the ASR by calculating $\frac{\text{\#predictions of the attacker targeted class}}{\text{\#total predictions}}$ on a modified version of the test set without samples from the attacker targeted class, after which we apply the trigger to all the samples.

Clean Data Accuracy (CDA) The Clean Data Accuracy indicates how well the backdoored model can correctly predict the class of test samples without a trigger[13]. We obtain the ASR by calculating $\frac{\text{\#correct predictions}}{\text{\#total predictions}}$ on the test set.

We declare a backdoor attack successful when the ASR is near 100 percent, with a CDA similar to the clean model.

4.7 Defenses

We evaluate our backdoor attacks against different well-known defenses. These are mainly developed for the image domain, so we reason if and how they can be adjusted to the tabular data domain. By investigating various types of defenses, we hope to give valuable insights into what will and will not work for a defense against tabular backdoor attacks. For our experiments on defenses, we will look at all our backdoor attacks from section 4.5 performed on TabNet trained on all three main datasets. We chose TabNet because its implementation simplifies the setup needed to perform the defenses. We used FT-Transformer instead for the Fine-Pruning defense because its mostly unmodified transformer architecture makes it easier to reason which layers to use for pruning. We leave testing on other models for future work.

4.7.1 Reverse engineering-based defenses

In general, reverse-engineering defenses work by reconstructing the trigger from the model and testing whether the reverse-engineered trigger is causing a backdoor by using a threshold on some metric. These defenses assume that the defender has access to the (possibly backdoored) model and has a set of clean samples to validate the network. This corresponds to the scenario where the defender outsourced training to a third party. This also means the defender cannot access the training data used to train the model.

A well-known example of this type of defense from the image domain is Neural Cleanse[48]. Neural Cleanse uses an optimization scheme to find potential triggers with the smallest amount of perturbation which causes all samples to be misclassified to a single label. For each potential trigger (one per label), it then decides using an outlier detection algorithm whether a potential trigger is significantly smaller than the others. If the resulting value exceeds some threshold, it is considered an actual trigger to a backdoor in the model.

Note that this outlier detection algorithm will not work when there are only two classes, as with the LOAN and HIGGS dataset we use, or when multiple backdoors exist for different labels[49]. However, this can be solved by using the detection algorithm proposed by Xiang et al.[49], which is shown to work in both cases.

To indicate whether such reverse engineering defense could be adapted to a tabular data setting, we use a brute-force reverse engineering approach that sweeps over all expected inputs (including the slightly out-of-bounds trigger values) of each feature. Then, we plot how the classification results change on a held-out test set for each value. We perform this on both a clean model and the poisoned model, enabling us to see how a poisoned model differs in classification behavior from a clean model. This should allow us to detect specific classification behavior for a poisoned model and reason if it could be detected using some method.

4.7.2 Spectral Signatures

Spectral Signatures[45] is a training sample filtering-based defense that attempts to detect and remove poisoned samples from the training set by using statistics on the latent representations of the input. This defense assumes the adversary can access the training data and knows the model architecture and training process. However, the adversary does not train the model. This means the defender can access the training data used to train the (possibly backdoored) model but does not have a set of clean samples as with the outsourced training scenario.

As this defense does not use specific characteristics of the input data type, it can be adapted to our tabular data setting without many changes. We chose to use the 64 neurons at the input of the fully connected layer at the end of the TabNet model as our latent representations since these neurons are the output of the (TabNet) encoder. Since the output of the encoder should contain the encoded representations of the input[2], we argue this is the ideal location to obtain the spectral signatures.

The defense is performed in the following way. For each input in our training data, we obtain the activation values of the neurons in our selected layer. Then, for each input, we calculate the correlation with the top right singular vector of all activations. Finally, we plot a histogram of the correlation values and mark all poisoned samples for verification. The defense is successful when there is a clear separation between the distributions of the clean and poisoned samples, as the defender could then remove the samples from the smaller distribution with the higher correlation values and retrain the model. As this removes the poisoned samples from the training set, the resulting model will not contain a backdoor. Therefore, we evaluate this defense based on this separation between clean and poisoned samples.

4.7.3 Fine-pruning

Fine-pruning[29] is a model reconstruction-based defense that attempts to remove the backdoor from a model by directly manipulating the model's weights. Similar to the reverse engineering-based defenses, this defense assumes that the defender has access to the (possibly backdoored) model and has a set of clean samples to validate the network. Again, this corresponds to the outsourced training scenario. Additionally, the defender is assumed to be capable of fine-tuning the model. The defense consists of two parts, pruning and fine-tuning.

The pruning part is based on the idea that backdoored inputs activate neurons which are usually not activated by clean inputs, as shown to be the case in the BadNets attack[19]. By removing (pruning) those neurons, we could remove the backdoor activations and therefore eliminate the backdoor. This is done by performing a forward pass on the model using a set of clean samples while recording neuron activations in specific layers. Then, the defender iteratively prunes neurons with increasing activation values while recording accuracy until the accuracy drops below a certain threshold. As the fine-pruning defense is initially designed for CNNs, the authors only prune in the last convolutional layer of the CNN, as pruning on later layers has a larger impact on the model's behavior.

The fine-tuning part is performed after the pruning has finished. Fine-tuning is needed, as a model could use the same neurons for backdoored inputs used by clean inputs[29]. Fine-tuning should allow the model to reduce the accuracy drop by pruning these neurons. Fine-tuning is performed using the set of clean samples until the network converges. This process should perform much quicker than training a model from scratch.

We decided to evaluate the fine-pruning defense on the FT-Transformer model. Its mostly unmodified transformer structure makes it easier to reason about the layers to select for pruning. To investigate whether this defense could work in our setting, we evaluated this defense on the simplest version of our backdoor attack; an out-of-bounds trigger with a trigger size of one on the Forest Cover Type dataset. As training and evaluation of the FT-Transformer model take significant time, we did not test against our different attack settings. As for the layers to apply pruning on, we decided only to prune the feed-forward layers, as these layers process the output of the self-attention layers. As we use a trigger size of one, the trigger is placed in a single feature. This means the self-attention layers will likely contribute little to the backdoor, as the trigger requires no context between features. Consistent with this theory, we found that the drop in accuracy required to remove the backdoor was doubled by also pruning the self-attention layers.

We iteratively prune neurons with increasing activation values on the clean test set until we remove the backdoor. Then, we use a relatively small set of 20 thousand clean samples to fine-tune the pruned model until convergence, of which 20 percent is used for validation.

4.7.4 Other types of defenses

Aside from the previously mentioned defenses, two other well-known categories of defenses for backdoor attacks are preprocessing-based defenses and testing sample filtering-based defenses.

Preprocessing-based defenses like Februus[9] use some form of preprocessing at the inference stage to change or remove a potential trigger such that it prevents a backdoor from being activated. We argue this type of defense can easily be averted by placing the trigger in the higher-importance features, as changes in these features can significantly drop the clean data accuracy. Therefore, we will not perform this type of defense.

Testing sample filtering-based defenses like STRIP[14] attempts to detect and filter poisoned inputs during inference time. STRIP, a well-known defense in this category, works by superimposing other samples onto the sample which we want to know is clean or poisoned. Poisoned samples can then be detected if they have relatively low randomness in their prediction probability. While this can work for images, superimposing samples from tabular data will likely completely remove all of our proposed triggers, rendering the defense useless. However, a similar defense strategy might work as long as the trigger features stay untouched. We leave the development of such defense to future work.

Chapter 5

Results

In this chapter, we present the results of our experiments, including the experimental setup for generating our results.

5.1 Experimental setup

Our experiments on the backdoor attacks were run on a Ubuntu 22.04 machine with two AMD Epyc 7302 16-core CPUs and 504GB RAM. Each model was trained on an Nvidia RTX A5000 or Nvidia RTX A6000, depending on which GPU was free, as the machine was shared with multiple users. Using this setup, training times for a single model during our experiments varied from several minutes to almost two hours, depending on the dataset and model. Our experiments on the backdoor defenses were performed on a Ubuntu 22.04 machine with a Ryzen 7 5800X CPU, 32GB RAM, and an Nvidia RTX 3050.

5.1.1 Model implementations

We used the following implementations of the three models used in this research. For TabNet, we used a widely used and actively maintained PyTorch implementation[1]. As the official implementation[2] was written in TensorFlow, we chose to use a PyToch implementation to keep changes in the code to a minimum when comparing to the different models. For FT-Transformer and SAINT, we used both the implementation published by the authors of their papers[3][4].

5.1.2 Model training and hyperparameter tuning

For all datasets, we split 20 percent of our test set for measuring the ASR and CDA. Of the resulting 80 percent, we then split 20 percent again for our validation set, which we use for hyperparameter tuning. The rest is used as training data. Properly tuning the hyperparameters of each model-dataset combination for the highest validation accuracy takes a long time because of the long training duration. Since the focus of this research is on backdoor attacks and not on obtaining the highest possible accuracy scores, we decided to instead use the tuned hyperparameters on the Forest Cover Type dataset from the model implementations in section 5.1.1 with minimal adjustments, as each of the implementations had tuned hyperparameters available for this dataset. We also used these hyperparameters

[1]`https://github.com/dreamquark-ai/tabnet/releases/tag/v4.0`
[2]`https://github.com/google-research/google-research/tree/master/tabnet`
[3]`https://github.com/Yura52/tabular-dl-revisiting-models`
[4]`https://github.com/somepago/saint`

for our other datasets while only changing the number of epochs based on the validation set, as the model performance was very close (within one or two percent) to what we would expect on these datasets. Only for TabNet, we had to lower the batch size and set the optimizer parameters to their default value to prevent inconsistent results between runs.

5.2 Trigger location

Our results on the trigger location are in two parts. First, we obtain feature importance scores and rankings, after which we use these results to investigate the change in ASR and CDA when we change the trigger location to features with different feature importances. Aside from the main three datasets, we also used a synthetic dataset with five informative and five uninformative features, all with similar feature distributions, as described in section 4.4.3.

5.2.1 Feature importance scores and ranking

As seen in B.1, we see consistent and similar feature importance rankings between classifiers for all four datasets. While the lower-importance features show less consistent rankings, their absolute scores are relatively similar. These results show that TabNet has similar feature importance rankings as decision trees, even though it is a transformer-based deep learning model. We also see that the four different decision tree-based classifiers show similar results. Since all tested models show similar results and our other two deep learning models, SAINT and FT-Transformer, have a relatively similar architecture as TabNet, we assume that the other two models, of which we cannot easily obtain feature importance scores, have comparable feature importance rankings.

Furthermore, we also observe that the feature importance scores for the LOAN dataset (see B.1.3) contain some outliers. This is expected with a dataset containing this many features. Still, we see similar rankings in the top and bottom five overall.

From the results on the SYN10 dataset (see B.1.4), we can see that all five classifiers have all the informative features in the top five rankings and all the uninformative features in the bottom five rankings. This confirms that the feature importance scores work as expected, resulting in higher values for more important features and lower values for less important features.

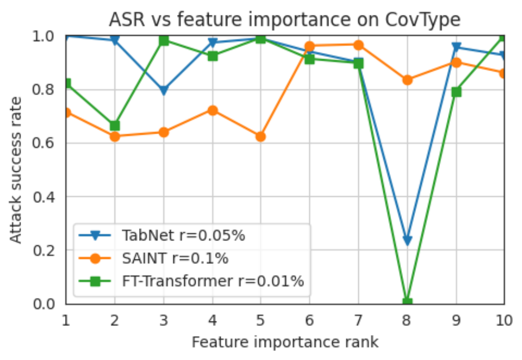
5.2.2 Feature importance vs. attack performance

We summarized our results in figure 5.1 by selecting the poisoning rate for each model-dataset combination that showed the most significant difference between different trigger locations. For the interesting full ASR plots, including a subplot with the feature distribution, we refer to appendix section B.2. For the worst-performing trigger position, a three percent poisoning rate allowed the model to achieve an attack success rate close to 100 percent. Therefore, with our datasets and models, the trigger position does not matter when the poisoning rate is three percent or higher.

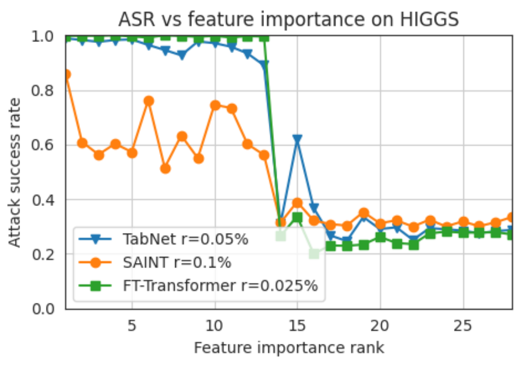
As seen in figure 5.1 and the figures in section B.2.2, the attack success rate can vary greatly depending on the trigger location. For example, figure B.3 shows that when the trigger is placed on feature "m_bb", FT-Transformer achieves an almost 100% ASR with a poisoning rate of 0.005% (which is only 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 higher. Therefore, we can conclude that trigger position can be crucial for a successful attack at low poisoning rates.

When we look at what determines the suitability of a feature to place the trigger on, we see the following. As seen in figure 5.1, only for the HIGGS dataset and our synthetic dataset do we see a clear relation between feature importance and ASR for all of our three models. We see no such relation for the other two datasets and, in some cases, even varying results between models. Interestingly, the relation could also be seen with the feature's distribution for the HIGGS dataset, as displayed in the bottom right corner of each plot in the figures in section B.2.2. Uniformly distributed features tend to perform worse as a backdoor trigger, as shown with the HIGGS dataset and the feature ”Aspect” from the CovType dataset (see figure B.2.1). On the other hand, features with a very tall and narrow distribution tend to perform better as a backdoor trigger, as shown in figure B.5. The reason for this behavior is unknown to us. When we look at the results on our synthetic dataset, where each feature has similar distributions and only differs in feature importance towards classifying the target label, we do see a clear relation between feature importance and ASR. From this, we conclude that there is at least a positive relation between feature importance and attack performance. However, we also conclude that there are more significant factors than just the feature importance that affects the suitability of a feature to place the trigger on.

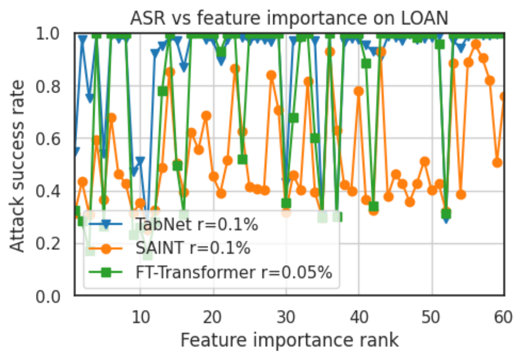
We found that the CDA stayed within run variance for all our trained models for all poisoning rates. Our low poisoning rates can explain this. Therefore, we only show the ASR in our results unless we see a change in the CDA.



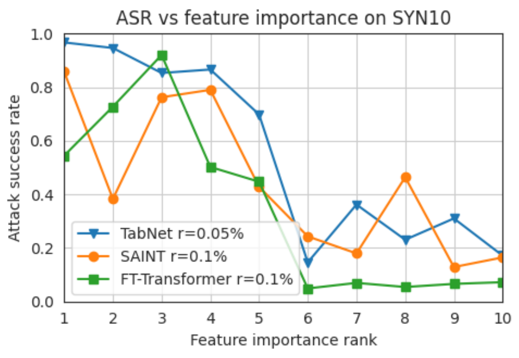
(a) Forest Cover Type dataset



(b) Higgs Boson dataset



(c) Lending Club Loan dataset



(d) SYN10 dataset

Figure 5.1: Change in ASR when the trigger position changes to features with lower feature importance rankings. Results are averaged over three runs.

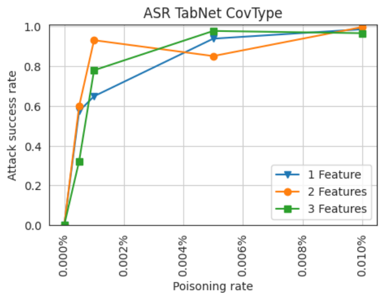
5.3 Trigger size

From our results in figures 5.2, 5.3 and 5.4, we generally see that larger trigger sizes require a lower poisoning rate for the same ASR, as we would expect. However, in most cases, the difference is relatively small. We would say only for SAINT and FT-Transformer on the CovType dataset, a larger trigger size greatly increases the performance of the attack. We see that the trigger size does not matter when using a poisoning rate of 0.5 percent or higher for almost all combinations of models and datasets. Only for SAINT on the CovType dataset, a trigger size of 1 does not reach an ASR of more than 99 percent, even with a poisoning rate of 1 percent.

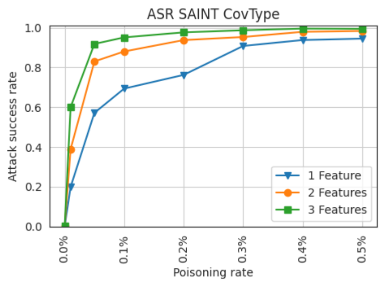
For the LOAN dataset and, to a lesser extent, also for the HIGGS dataset, we see that a smaller trigger size achieves a higher ASR for extremely low poisoning rates. We do not see this for the CovType dataset. This is because we use high-importance features for our trigger. By setting these features to a relatively high value, we influence the predictions from a benign model, as single features can greatly impact the output prediction. As the tasks on these two datasets are binary, the ASR includes all samples that the benign model misclassifies towards the target class. Therefore, we expect an ASR of $1 - (\text{CDA})$ on a clean model without any trigger. However, looking at the ASR at 0 percent, we see increasingly lower ASR for larger trigger sizes. In this case, the trigger values are causing the models to classify towards the non-target class, which is why we see a lower ASR for larger trigger sizes. This explains why we see lower ASR for larger trigger sizes until the model learns the trigger, after which we see better ASR for larger trigger sizes. We do not see this in the results for the CovType dataset, as the tested models obtain a very high CDA while also being a multi-class classification problem. However, we can still observe the same effect by looking at the confusion matrix on the ASR test set for different trigger sizes, including no trigger, as seen in figure 5.5. From there, we see that with no trigger on a benign model, the model predicts the samples similar to the distribution of the clean labels. When we add the trigger, we see that the model starts to favor only one or two classes.

Another observation we can make from our results is that, in general, out of the three tested models, FT-Transformer shows the highest attack success rates given a poisoning rate. TabNet shows quite similar results but often slightly worse. SAINT performs by far the worst out of the three models. We argue that SAINT performs the worst because of its row attention mechanism. As our poisoned samples are randomly spread throughout the dataset, the row attention mechanism will likely not learn any backdoor trigger. Furthermore, row attention is designed to improve performance by allowing the model to borrow a feature from similar samples in the same batch when it detects a noisy or missing value[42]. Our trigger can be seen as noisy features, explaining the lower attack success rate. We verify this theory by running SAINT with only column attention on the CovType dataset with a trigger size of 1. As seen from figure 5.6, we see that compared to SAINT with row attention, it achieves a much higher attack success rate for the same poisoning rate. However, removing row attention reduces the accuracy by two percent, as expected, since the row attention is designed to improve performance on clean data.

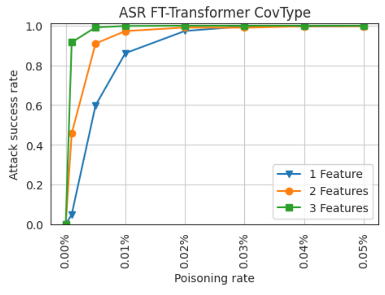
We argue that TabNet performs slightly worse than FT-Transformer because of its feature selection mechanism and smaller model size. These properties should cause the model to be less prone to overfitting, making it less likely to learn a backdoor.



(a) TabNet

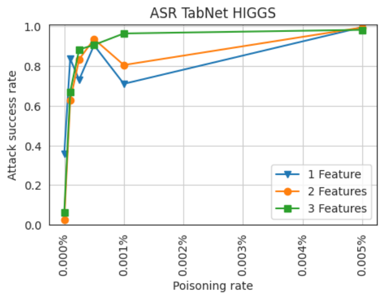


(b) SAINT

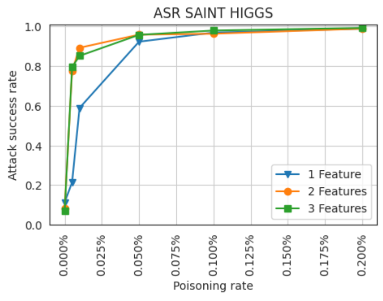


(c) FT-Transformer

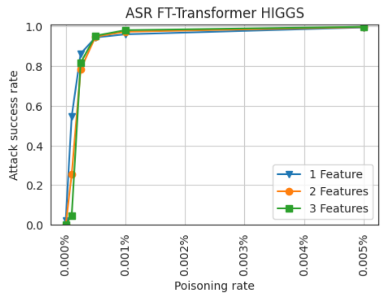
Figure 5.2: ASR for different trigger sizes on the Forest Cover Type dataset, averaged over five runs



(a) TabNet

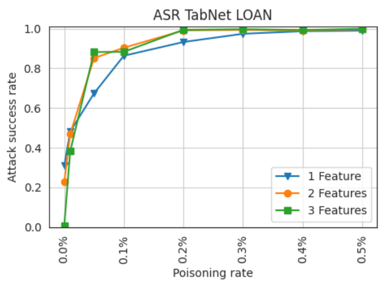


(b) SAINT

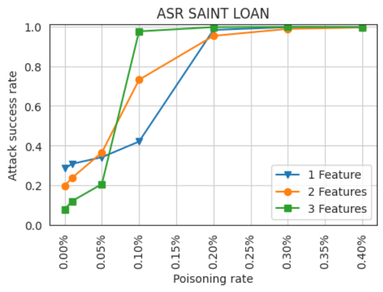


(c) FT-Transformer

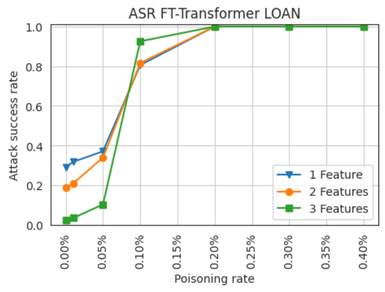
Figure 5.3: ASR for different trigger sizes on the Higgs Boson dataset, averaged over five runs



(a) TabNet

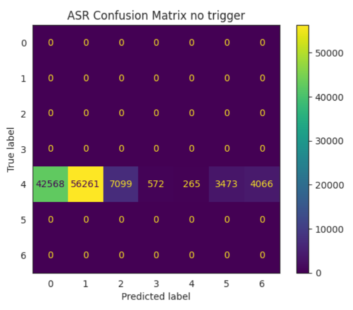


(b) SAINT

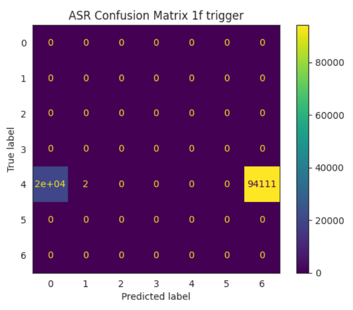


(c) FT-Transformer

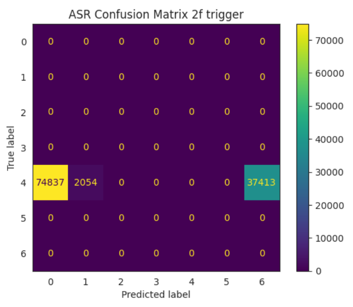
Figure 5.4: ASR for different trigger sizes on the Lending Club LOAN dataset, averaged over five runs



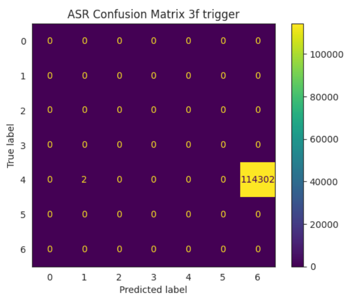
(a) No trigger



(b) 1 Feature



(c) 2 Features



(d) 3 Features

Figure 5.5: Confusion matrices for different trigger sizes on ASR test set (true label is target label) for benign TabNet model on Forest Cover Type dataset

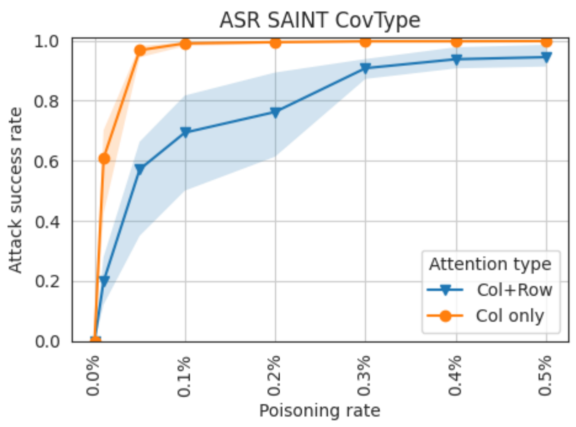
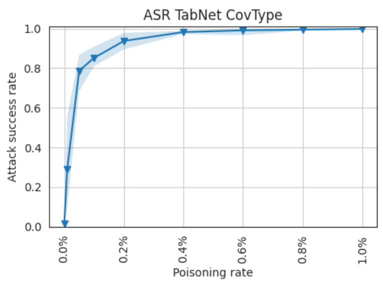


Figure 5.6: ASR for SAINT with and without row attention on the CovType dataset with a trigger size of 1 and out-of-bounds trigger value, averaged over five runs

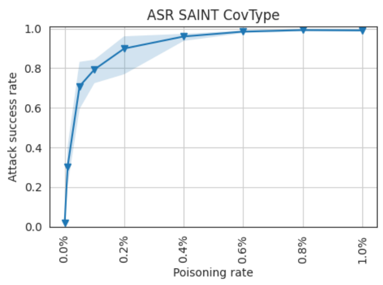
5.4 In-bounds trigger value

From our results in figures 5.7, 5.8 and 5.9, we see that using the most common value in the training set for each trigger feature still results in a working attack, although requiring a larger poisoning rate of up to three percent.

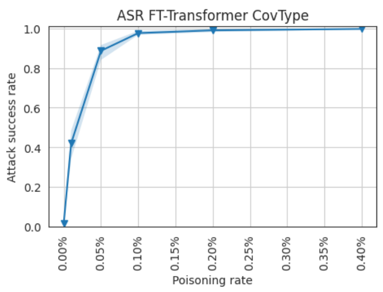
Interestingly, by using this type of trigger on the HIGGS dataset, the clean models (indicated with a 0 percent poisoning rate) already predict the target class for up to 90 percent of the test set, causing a very high ASR at 0 percent. This behavior is similar to section 5.3 except that the trigger now represents a sample of the target class. Still, we need up to a hundred times more poisoning than the out-of-bounds trigger (see figure 5.3) to obtain a near-perfect ASR.



(a) TabNet

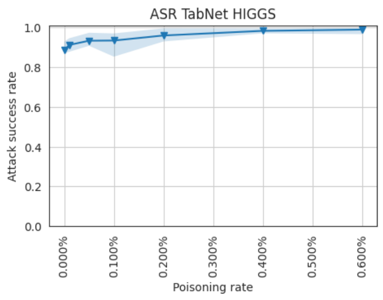


(b) SAINT

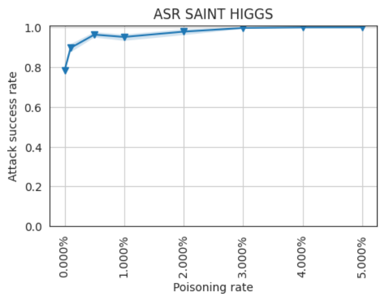


(c) FT-Transformer

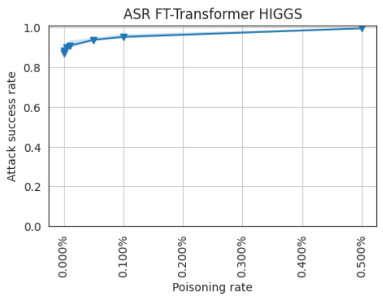
Figure 5.7: ASR for in-bounds trigger value with trigger size of 3 on the Forest Cover Type dataset, averaged over five runs including min-max error band



(a) TabNet

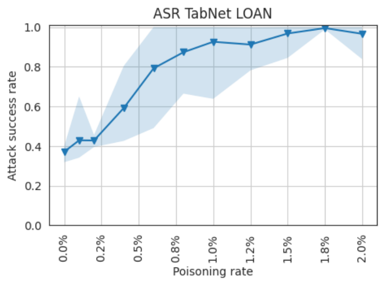


(b) SAINT

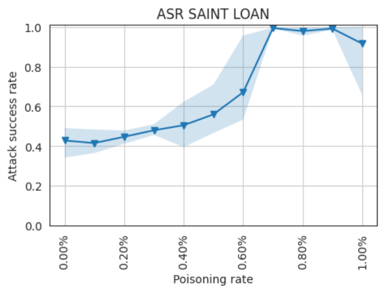


(c) FT-Transformer

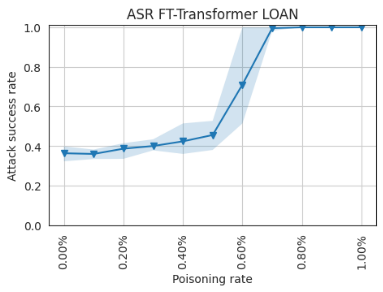
Figure 5.8: ASR for in-bounds trigger value with trigger size of 3 on the Higgs Boson dataset, averaged over five runs including min-max error band



(a) TabNet



(b) SAINT



(c) FT-Transformer

Figure 5.9: ASR for in-bounds trigger value with trigger size of 3 on the Lending Club LOAN dataset, averaged over five runs including min-max error band

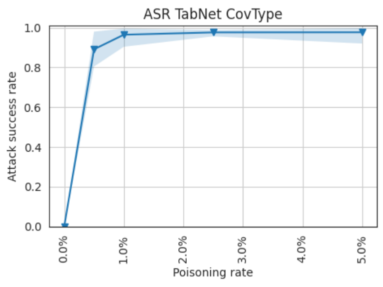
5.5 Clean label attack

From our results in figures 5.7, 5.8 and 5.9, we see that in all cases except for SAINT on the CovType dataset, the clean label attack is successful. When we compare the clean label attack against the dirty label attack with the same trigger size of one (see figures 5.2, 5.3 and 5.4), we see the following.

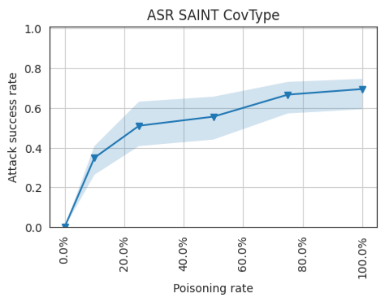
As expected, we need a higher required poisoning rate with the CovType dataset than the dirty label attack. This is mainly because the poisoning rate is calculated over the samples it can poison, which in the clean label attack setting are only the samples with the target label. As the target class only has 6075 samples in the training data, a poisoning rate of 1 percent in the clean label attack settings means that about 60 samples are poisoned. If we look at the number of poisoned samples, then this corresponds to a poisoning rate of 0.016 percent in the dirty label attack. When we look at the results of TabNet in figure 5.2a, we see that for a dirty label attack, we need about 0.01 percent of poisoning for a successful attack. This equals 37 samples. As seen in figure 5.10a, for our clean label attack, we need about 2 percent of poisoning for a successful attack. This equals 121 samples. Therefore, for TabNet on the CovType dataset, we need more than three times the amount of poisoning for a clean label attack to achieve the same ASR. An increase in required poisoning is expected, as the model can learn from a sample by either their trigger or the other features, which are now informative enough to classify the target label. In a dirty label attack setting, the model is forced to look only at the trigger, as the other features do not match the label.

For both the HIGGS and LOAN datasets, we see about the same attack success rates for double the poisoning rates compared to the dirty label attack. Since both datasets are well-balanced, the clean label attack requires the same amount of poisoning in terms of the number of samples.

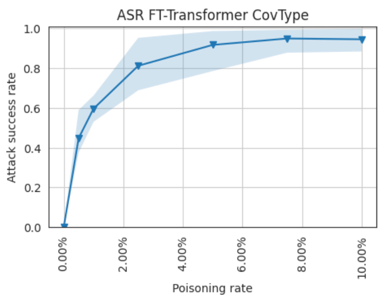
As for SAINT on the CovType dataset, we find that there are not enough samples in the target class for the backdoor attack to work, as the attack success rate only reaches 75 percent at a poisoning rate of 100 percent. With such high poisoning rates, we see a drop in clean data accuracy, as shown in figure 5.13. We see that the clean data accuracy starts to decline only after 50 percent of poisoning. We note that this was the only experiment where we found a drop in clean data accuracy.



(a) TabNet

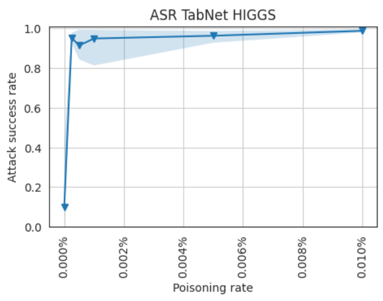


(b) SAINT

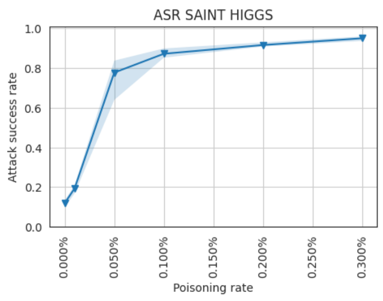


(c) FT-Transformer

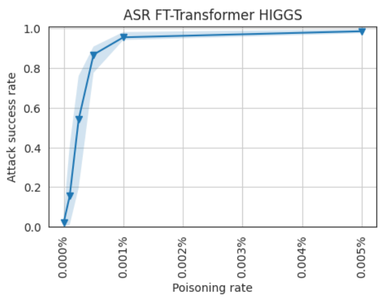
Figure 5.10: ASR for clean label attack with trigger size of 1 and out of bounds trigger value on the Forest Cover Type dataset, averaged over five runs including min-max error band



(a) TabNet

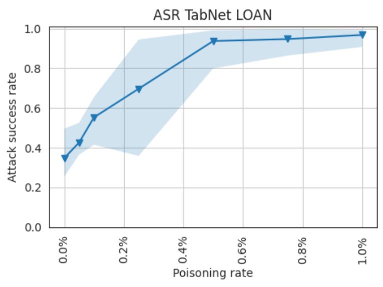


(b) SAINT

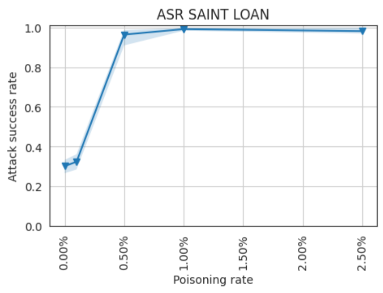


(c) FT-Transformer

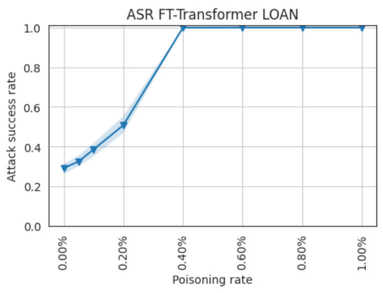
Figure 5.11: ASR for clean label attack with trigger size of 1 and out of bounds trigger value on the Higgs Boson dataset, averaged over five runs including min-max error band



(a) TabNet



(b) SAINT



(c) FT-Transformer

Figure 5.12: ASR for clean label attack with trigger size of 1 and out of bounds trigger value on the Lending Club LOAN dataset, averaged over five runs including min-max error band

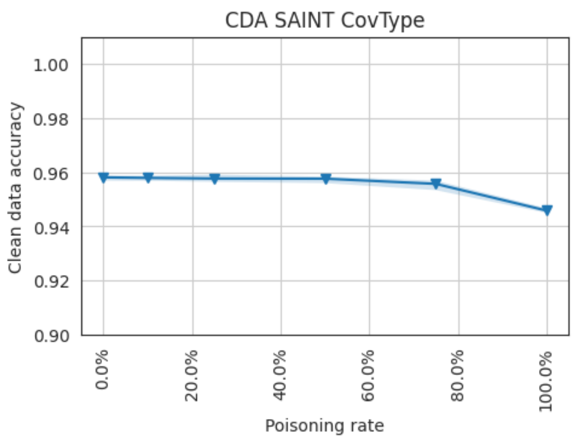


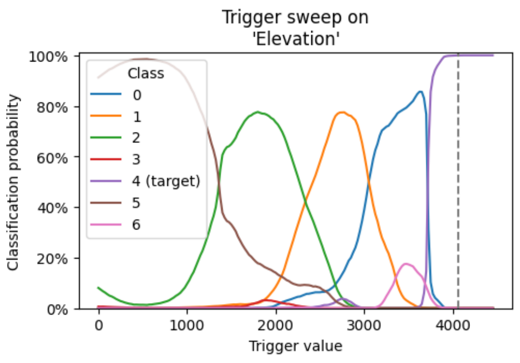
Figure 5.13: CDA for clean label attack with trigger size of 1 and out of bounds trigger value on the Forest Cover Type dataset using SAINT, averaged over five runs including min-max error band

5.6 Defenses

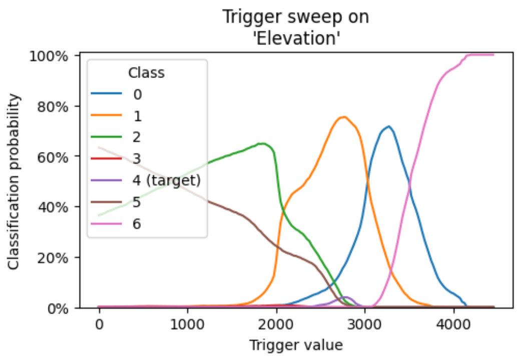
5.6.1 Trigger reverse engineering

Depending on the trigger location, the classification results from the backdoored model can be indistinguishable from a clean model when the target label is unknown. This can be seen in figure 5.14 on the Forest Cover Type dataset, where high values for the high importance feature "Elevation" causes the clean model to always predict class 6 in a very similar way as the backdoored model always predicts the target class 4. Furthermore, even for the poisoned model, we see a 100 percent classification rate towards non-target class 5 for values around 500, indicating a backdoor. These results relate to the challenges of tabular data (see section 4.1), where especially high-importance single features can significantly influence the output prediction. When we instead look at a trigger using a low-importance feature like "Slope" (see figure 5.14), we do not see any false positive backdoors anymore, which means that a reverse engineering defense could detect the trigger in this case, assuming a trigger size of one.

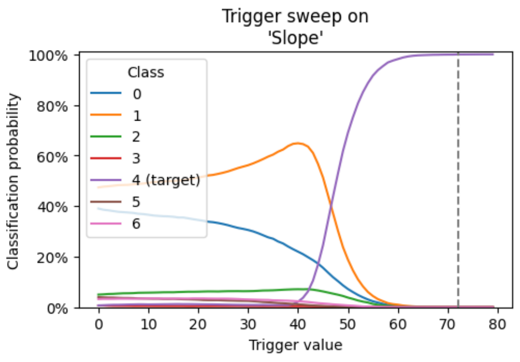
Looking at larger trigger sizes, we find that a smaller subset of the trigger is not always enough to achieve a high attack success rate, as seen in figure 5.15. In this case, a single change in one of the two trigger features cannot trigger the backdoor entirely. However, when we look at the "sub_grade" feature in the clean model, a large value similar to the trigger value of the poisoned model would cause the classifier to classify more than 90 percent of the samples towards one class. We argue that a reverse engineering algorithm is more likely to find this as a potential trigger candidate than the actual trigger, as a smaller perturbation is needed.



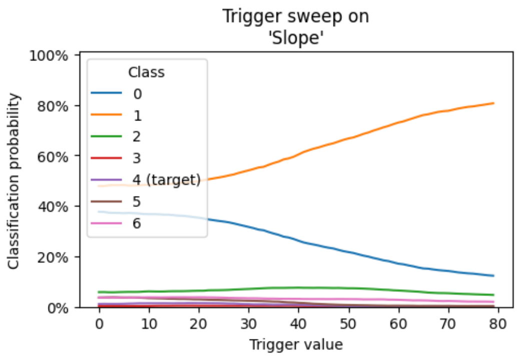
(a) Backdoored TabNet model with "Elevation" trigger



(b) Clean TabNet model ("Elevation")

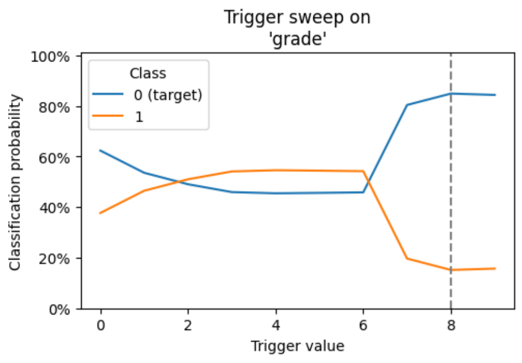


(c) Backdoored TabNet model with "Slope" trigger

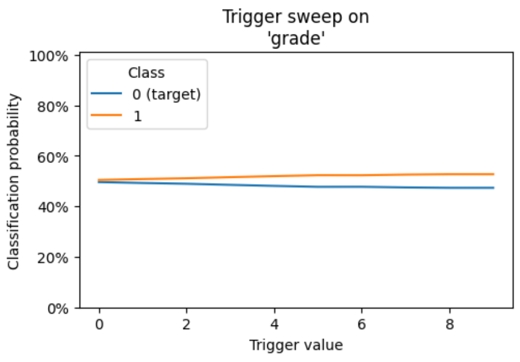


(d) Clean TabNet model ("Slope")

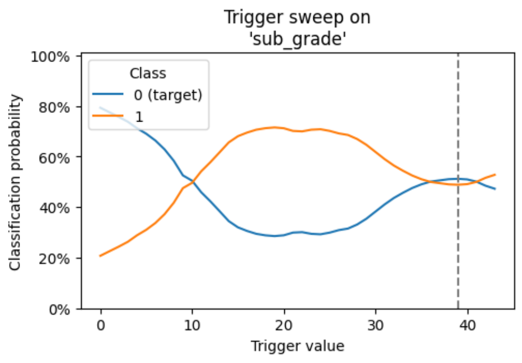
Figure 5.14: Classification probabilities on the test set of the Forest Cover Type dataset for different potential trigger values of the high importance feature "Elevation" and low importance feature "Slope". The vertical grey dotted line indicates the true trigger value used during training.



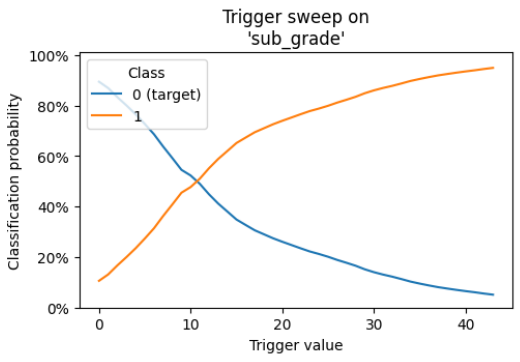
(a) Backdoored TabNet model with trigger size 2 ("grade")



(b) Clean TabNet model ("grade")



(c) Backdoored TabNet model with trigger size 2 ("sub_grade")



(d) Clean TabNet model ("sub_grade")

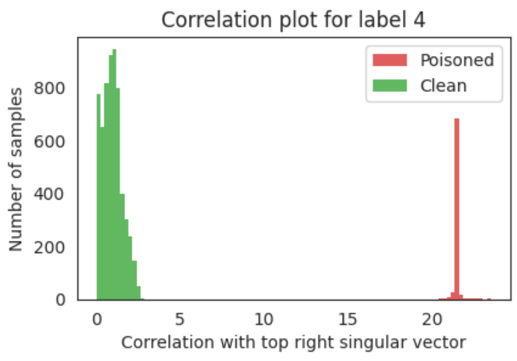
Figure 5.15: Classification probabilities on the test set of the Forest Cover Type dataset for different potential trigger values of the high importance feature "Elevation" and low importance feature "Slope". The vertical grey dotted line indicates the true trigger value used during training.

5.6.2 Spectral Signatures

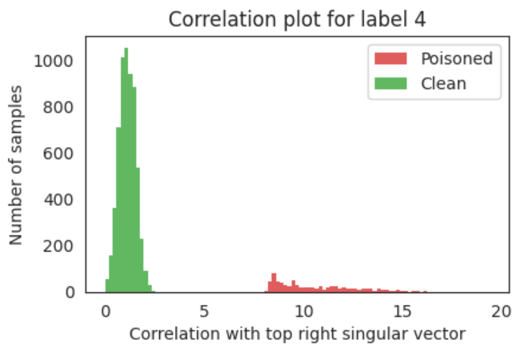
As seen in figures 5.16, 5.18 and 5.19, a clear separation between the poisoned samples and the clean samples can almost always be found, indicating that this defense is quite effective. Only for the in-bounds trigger on the HIGGS dataset, there is no distinction between poisoned and clean samples. We argue this is the case because of the following. 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 5.4, the backdoored model will likely not have drastically different activations for poisoned samples, resulting in similar distributions. Furthermore, we see a less clear separation between the two distributions for the in-bounds trigger on the Forest Cover Type dataset. We argue the in-bounds trigger is more difficult to detect by this defense as the individual trigger values now correspond to clean samples, which may cause more similar neuron activations. Still, there is a large enough separation to remove enough poisoned samples to render the backdoor ineffective.

In one case, we found a false positive for label 6 in the CovType dataset while using a trigger size of one. As seen in figure 5.17, a second distribution on the right side indicates poisoned samples. However, as the target label is 4, these samples are all clean. We only found this false positive when using the low-importance feature "Slope" as the trigger. After further inspection of the samples of both distributions, we found that the seemingly poisoned distribution contained all samples with a higher "Elevation" feature value (all larger than 3639) and similar one-hot encoded feature values. As shown in the trigger sweep on a clean model on the "Elevation" feature (see figure 5.14), a high "Elevation" feature value behaves similarly to a backdoor trigger with target label 6. This, together with the same one-hot encoded features, could cause the model to learn these samples similar to a backdoor trigger, causing different activations resulting in this separate distribution. Why this only happened a single time could be explained by variations in learning behavior between runs, as these samples might not always make a clear enough trigger for the model to learn.

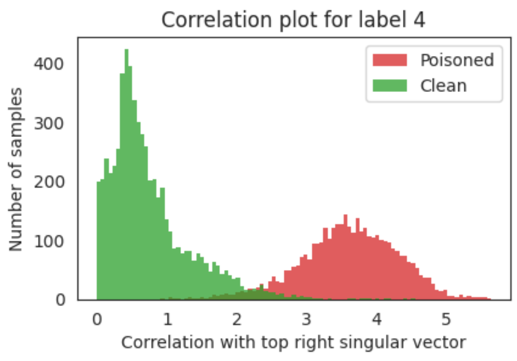
Finally, as seen in figure 5.16, the clean-label attack can still be defended against using spectral signatures, as there still is a clear separation between the two distributions. It appears that, even though the poisoned samples are more in line with the clean samples, the model still has different neuron activations for the poisoned samples.



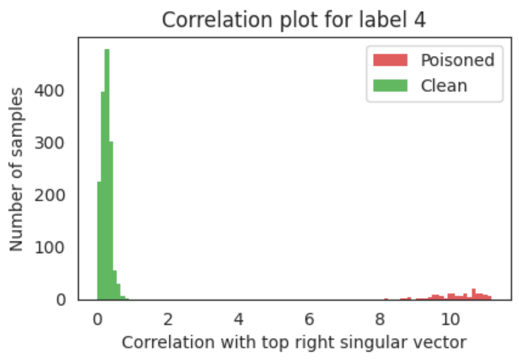
(a) Trigger size 1



(b) Trigger size 3



(c) In-bounds trigger



(d) Clean label attack (Trigger size 1)

Figure 5.16: Correlation plots for TabNet trained on the Forest Cover Type dataset

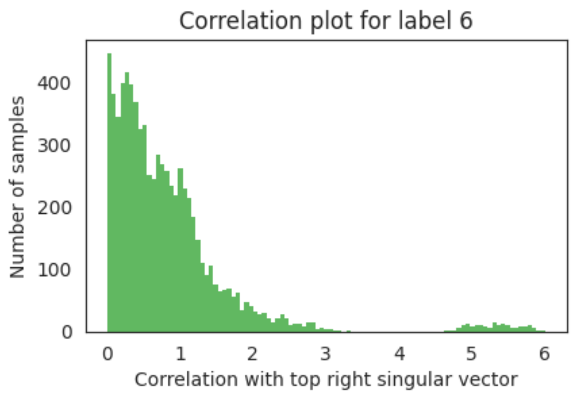
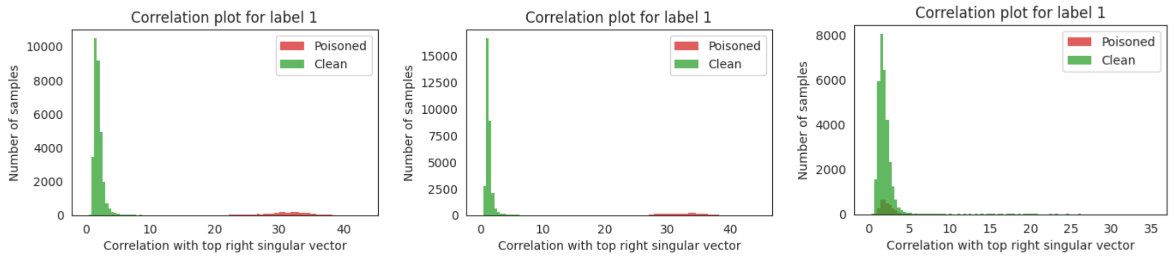
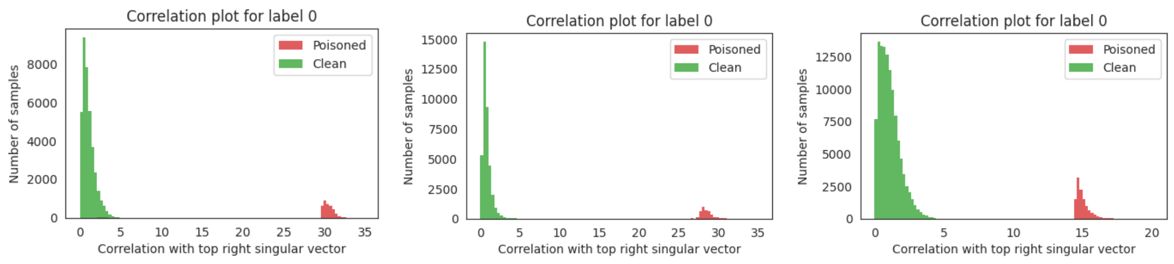


Figure 5.17: Correlation plot for non-target label 6 for TabNet trained on the Forest Cover Type dataset



(a) Trigger size 1 (b) Trigger size 3 (c) In-bounds trigger

Figure 5.18: Correlation plots for TabNet trained on the Higgs Boson dataset



(a) Trigger size 1 (b) Trigger size 3 (c) In-bounds trigger

Figure 5.19: Correlation plots for TabNet trained on the Lending Club Loan dataset

5.7 Fine-Pruning

As seen in figure 5.20, we needed to prune 50 percent of the output neurons in the feed-forward layers to remove the backdoor from the network. Over this 50 percent, we see a gradual decline in clean data accuracy, while the attack success rate only starts to decline after around 20 percent of the neurons are pruned. This behavior is similar to the results of a pruning-aware attack on CNNs[29], indicating that in the transformer model, the backdoor activations occur in the same neurons as clean activations.

As seen in table 5.1, after removing the backdoor, the clean data accuracy dropped from 95.4 percent to 70.2 percent, while the attack success rate dropped from 99.7 percent to 1.7 percent. By fine-tuning the pruned network, we increased the clean data accuracy to

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

Table 5.1: Results of the fine-pruning defense on FT-Transformer on the CovType dataset

92.1 percent, while the attack success rate remained low at 4.2 percent. Therefore, the fine-pruning defense was able to remove the backdoor with a clean data accuracy drop of 3.3 percent, assuming the defender stops pruning right after the backdoor is removed.

As for the training times, the time to train the model from scratch was one hour and 17 minutes, while the fine-tuning process only took two minutes and 30 seconds.

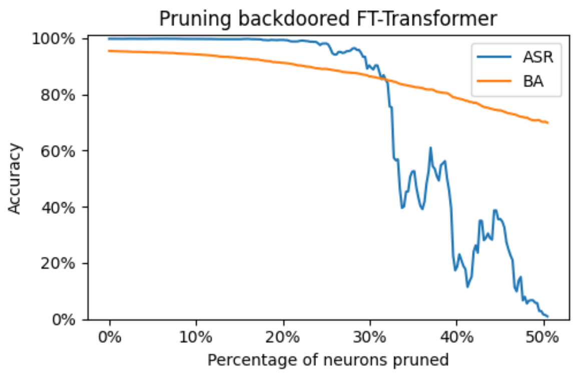


Figure 5.20: Pruning results of fine-pruning defense on FT-Transformer using a single feature trigger on the Forest Cover Type dataset.

Chapter 6

Discussion

In this chapter, we further discuss the results from chapter 5 and present possible future work directions.

6.1 Backdoor attacks

In this section, we discuss the backdoor attacks performed in this thesis.

6.1.1 Trigger location

One of our major findings in this research is the importance of trigger position. We found the difference in the required poisoning rate for a successful attack can be at least twenty times higher depending on the trigger position. While we do not fully understand what makes one trigger position more successful than the other, we did find a positive relationship between feature importance and attack performance. This finding contradicts the results of Xie et al.[51], who show a negative relation between feature importance and attack performance. This can be explained by the following. Xie et al. only performed their attack on the LOAN dataset. When we look at our results from the LOAN dataset (see section B.2.3), we see that for this particular dataset, especially for SAINT and FT-Transformer, the lowest importance features tend to achieve a higher ASR than the highest important features. As their results only show two data points, we can see how their results show a negative relation between feature importance and ASR. However, when we look at all 60 numerical features, we do not see any relation between feature importance and ASR on this dataset.

We also argue that using high-importance features as the trigger location makes the attack harder to defend against. Our trigger reverse-engineering experiments show that classification behavior from a backdoored model using a high-importance feature often cannot be distinguishable from a clean model. This was not the case for a backdoored model using a low-importance feature, making the latter arguably less stealthy. Furthermore, a simple defense against backdoor attacks using low-importance features would be to remove the low-importance features as they contribute the least towards the model performance.

6.1.2 Trigger size

From our experiments on the effect of the trigger size, we generally saw an increase in the attack success rate for larger trigger sizes. While this increase is expected, we found it to be relatively small compared to results from the image domain[1]. We argue this increase in trigger size is less significant because of how tabular data works compared to image data. In contrast to image data, tabular data generally consists of relatively few features while being more informative on their own. Our results show that a single feature is usually enough to achieve a near-perfect ASR while requiring only a low poisoning rate. With image data, single features are less informative as much of the information is encoded by the spatial dependencies between features.

6.1.3 In-bounds trigger value

We found our backdoor attacks using an in-bounds trigger value to require a significantly higher poisoning rate to achieve the same attack success rate. 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 cause the backdoor to activate.

While the required poisoning rate is higher than an out-of-bounds trigger, the in-bounds trigger still has several advantages. As the trigger values are in-bounds, all features, including nominal ones, can now be used as a trigger without invalid values. Furthermore, we argue that using in-bounds trigger values makes poisoned samples less likely to be detected by humans and tools such as outlier detection algorithms.

6.1.4 Clean label attack

As expected, we saw on the Forest Cover Type dataset an increase in the amount of poisoning needed to achieve the same ASR compared to our dirty-label attack. However, this increase was less than we initially expected as we achieved a near-perfect ASR in most cases, indicating that the attack still works. In contrast to the image domain[46], we did not have to adapt our poisoning strategy to suppress the features related to the target class to force the model to learn from just the trigger. However, the attack might become ineffective if the target class only contains a small number of samples, as can be the case with an imbalanced dataset.

Interestingly, we found that the clean label attack achieved similar attack success rates with similar amounts of poisoned samples for both of our binary classification datasets. We argue the difficulty of the classification task can explain this, as our models can only achieve 75 and 67 percent accuracy on these datasets. Therefore, the features are likely less informative, which makes them less of a "distraction" from learning the trigger. Additionally, the trigger overrides the most important feature with the trigger value, further decreasing the quality of the sample.

6.2 Backdoor defenses

In this section, we discuss the backdoor defenses performed in this thesis.

6.2.1 Spectral signatures

As shown in section 5.6.2, the Spectral Signatures defense successfully defended against most of our backdoor attacks in various settings, with only one case of a false positive. From these results, we argue defenses using the assumptions of separability of poisoned and clean samples in latent space, like Spectral Signatures but also others like Activation Clustering[7], SCAn[43], and SPECTRE[20], can likely be used to defend against backdoor attacks similar to ours. However, this type of defense is not an end-all solution to our backdoor attacks, as it has several disadvantages. For one, this type of defense is likely not applicable to the outsourced training scenario as the attacker is unlikely to provide the poisoned samples to the user. Furthermore, retraining the entire model from scratch after removing the poisoned samples can be quite expensive, depending on the size of the model and data.

6.2.2 Trigger reverse engineering discussion

From the results in section 5.6.1, we argue that a reverse engineering defense cannot be easily adapted to the tabular data domain. The main reason is that a change in a single high-importance feature can greatly influence the output predictions, making this change challenging to distinguish from a backdoor trigger. Especially triggers with larger trigger sizes are more likely to go undetected by this defense method, as a trigger reverse-engineering algorithm would more likely find a smaller potential trigger in one of the high-importance features.

We expect this type of defense to work slightly better for datasets with more balanced feature importance scores since, in this case, other features could influence the model enough to prevent convergence towards a single output label. Still, we do not see this type of defense to be successful until the issues mentioned above are solved.

6.2.3 Fine-Pruning

As shown in section 5.7, the fine-pruning defense was able to defend against our backdoor attack at the cost of a small drop in accuracy. One of the main advantages of the fine-pruning defense is that it can be applied without requiring access to poisoned samples, as is required with Spectral Signatures, allowing it to be applied in more scenarios. However, this defense also has a considerable disadvantage. We found the pruning behavior similar to a pruning-aware attack on CNNs, indicating the absence of backdoor-specific neurons. This absence of backdoor-specific neurons means it is unclear for the defender when to stop pruning, as the defender cannot stop when the clean data accuracy starts to drop. While we could verify the attack success rate in our experiment for verification purposes, the defender cannot. To the best of our knowledge, this issue has not been discussed in the original paper. As the defender has no indication of the presence of a backdoor, and as the defender has no indication whether a potential backdoor has been removed, we do not think this defense can be applied in many scenarios. However, if this defense can consistently remove backdoors with a better pruning strategy before a clean data accuracy drop occurs after fine-tuning, this defense could become a more viable solution.

6.3 Limitations

A limitation of this work is the lack of a proper understanding of the perceivability of perturbations on tabular data. For images, the stealthiness of an attack can be defined as the pixel-wise distance between the clean and poisoned sample[27]. For tabular data, it is unclear what stealthiness means in this domain. Humans look differently at tabular data than at images or text. How humans perceive tabular samples is not intuitive since tabular data (in the context of machine learning) is primarily made for machines. As far as we know, there is currently no clear consensus on a metric to define the perceivability of perturbations on tabular data[17][3][31].

As we do not know how to measure the perceivability of perturbations in tabular data, we cannot properly define a metric that measures the stealthiness of our attacks. Furthermore, it is difficult to say what defines too many perturbations, especially when perturbing high-importance features, since these perturbations quickly change the predicted class on a clean model. An extreme example would be a trigger that changes all features to values of a sample of the target class. While an attack using this trigger would clearly not be considered a backdoor attack, it is difficult to draw a line.

6.4 Future work

This section provides possible future research directions related to or extending our work.

6.4.1 Further research on attack parameters and variations

To better understand the perceivability of perturbations in tabular data and, by extension, the stealthiness of backdoor triggers in tabular data, further research is required. We suggest performing a user study to gather more insights into what causes humans to detect various backdoor triggers in tabular data.

We also argue it is important to better understand the trigger position parameter of the backdoor attack. As shown in section 5.2, we found a positive relation between feature importance and attack success rate. However, the results also show that something other than feature importance determines the effectiveness of a trigger location. This is important to research, as depending on the feature selected, one might need an order of magnitude larger poisoning rate to achieve the same ASR.

Finally, we argue a sample-specific trigger generation method could further increase the stealthiness of the backdoor attacks. In contrast to our presented trigger generation methods, a sample-specific method could account for mutual exclusive feature values between trigger and non-trigger features.

6.4.2 Further research on defenses

As stated in section 4.7, we attempted to give valuable insights into the potential of several defense directions. As we have shown the effectiveness of our backdoor attacks, we hope to see the development of a defense specifically for tabular data.

One of the defenses we used we would suggest investigating further is fine-pruning, as there are still more directions to explore for this defense. For example, to better understand this defense's generalizability, we suggest performing fine-pruning on different models, datasets, and trigger generation methods. Furthermore, depending on the model used, different pruning strategies (e.g., what layers to prune) could be explored. For example, with larger trigger sizes, it might also make sense to prune the multi-head attention layers.

Another direction we suggest for future work is creating an adaptive attack for Spectral Signatures since it successfully defended against most of our attacks. There already exist various works on how to decrease the effectiveness of such defense[37][10][38]. Many of these[10][38] work by adding some form of regularization in the loss function of the training process to reduce the separability between poisoned and clean samples in latent space. However, this changes the threat model as the attacker now also needs control over the training process. While this is a realistic scenario, as it corresponds to the outsourced training scenario, it greatly reduces the defense's applicability. This is because this type of defense assumes the defender has access to the training data used to train the backdoored model. However, in an outsourced training scenario, the attacker will unlikely send the poisoned samples to the user.

Recently, work by Qi et al.[37] shows how to perform regularization by applying a specific data poisoning strategy. They could circumvent the well-known latent separability-based defenses without modifying the training process. As this attack strategy was developed for image data, it would be interesting to see whether a similar strategy can be applied to tabular data.

6.4.3 Other settings for machine learning using tabular data

To stay close to other research on backdoor attacks, we chose only to investigate backdoor attacks on classification tasks. However, regression tasks are also quite common with tabular data. Furthermore, the models we tested are all capable of handling regression tasks. It would be interesting to see how susceptible these models are on regression tasks and what backdoor strategy could be applied to this type of task.

Another setting we think is interesting for future work is backdoor attacks on classical machine learning algorithms. While classical machine learning algorithms like XGBoost have smaller data and training requirements, backdoor attacks on these models could still present a risk in certain scenarios where the attacker might influence part of the training data. Therefore, seeing how these attacks perform on classical machine-learning models could be interesting.

Chapter 7

Conclusion

In this thesis, we showed that transformer-based DNNs for tabular data are vulnerable to backdoor attacks.

We found it non-trivial to adapt trigger-generation methods from other domains, as tabular data differs greatly from image or text data. The main challenges of creating a trigger for tabular data are the heterogeneous structure, lack of spatial dependencies between features, and single feature importance of tabular data.

By only changing one feature to a slightly out-of-bounds value, we could create successful backdoor attacks on all of the three tested models and datasets. Compared to the image and text domain, our attacks require relatively low amounts of poisoning to achieve high attack success rates. These high attack success rates using a small perturbation of only a single feature, together with the small amount of poisoning required, show how susceptible transformer-based models for tabular data are.

We found the trigger position to be crucial for the attack performance. Using a synthetic dataset with similar feature distributions, we found that high-importance features generally perform better than low-importance features. However, there also appear to be other factors determining what makes a feature suited for the trigger. Furthermore, we found a larger trigger size to increase the attack performance. However, we found this to be a less important parameter than backdoor attacks in the image domain. Finally, we showed that the model architecture can also greatly influence the attack performance. By including row attention in the transformer model, the attack performance is severely degraded.

While requiring an increased amount of poisoning, a more stealthy version of the attack using in-bounds trigger values also performed well on all of our tested models and datasets. We also found a clean-label attack to be surprisingly successful in most cases. In contrast to the image domain, we did not have to apply any method to force the model to learn from the trigger instead of the other features.

Aside from backdoor attacks, we also investigated the performance of several types of defenses against our backdoor attacks. We found many types of defenses not very successful against our attacks. Without drastic changes, we found preprocessing-based defenses like Februus and testing sample filtering-based defenses like STRIP non-applicable on tabular data. Reverse engineering-based defenses like Neural Cleanse are challenging to use on tabular data because of the single feature importance in tabular data. While Fine-pruning, a model reconstruction-based defense, was able to remove our backdoor successfully, it has several challenges making it hard to recommend in its current state. For one, we found the critical neurons used in the backdoor activation were also used in predictions on clean samples. Therefore, the backdoor cannot be removed without decreasing the clean data accuracy. Even when the defender stops to prune at the exact moment when the backdoor is removed, which they cannot know, the clean data accuracy is still over three percent lower after fine-tuning than before applying the defense.

The only successful defense against our attacks was Spectral Signatures, a training sample filtering-based defense that uses the separation between poisoned and clean samples in latent space. For almost all attack variations, it could create a clear separation between the poisoned and clean samples. Assuming the defender has access to the training dataset, which is less likely in an outsourced training scenario, this type of defense can successfully defend against our backdoor attacks.

As DNNs become more competitive in tabular data, the threat of backdoor attacks in tabular data also increases. We have shown how susceptible these models are to backdoor attacks and how hard these attacks can be defended against, especially in an outsourced training scenario. We conclude that more research into both backdoor attacks and defenses in the tabular domain is needed to understand the backdoor attacks better, including their stealthiness, and to develop effective strategies to defend against these attacks.

Bibliography

[1] Gorka Abad, Jing Xu, Stefanos Koffas, Behrad Tajalli, and Stjepan Picek. A systematic evaluation of backdoor trigger characteristics in image classification. *arXiv preprint arXiv:2302.01740*, 2023.

[2] Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, May 2021.

[3] Vincent Ballet, Xavier Renard, Jonathan Aigrain, Thibault Laugel, Pascal Frossard, and Marcin Detyniecki. Imperceptible adversarial attacks on tabular data. *arXiv preprint arXiv:1911.03274*, 2019.

[4] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105. IEEE, 2019.

[5] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

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

[9] Bao Gia Doan, Ehsan Abbasnejad, and Damith C Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. In *Annual Computer Security Applications Conference*, pages 897–912, 2020.

[10] Khoa Doan, Yingjie Lao, and Ping Li. Backdoor attack with imperceptible input and latent modification. *Advances in Neural Information Processing Systems*, 34:18944–18957, 2021.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[12] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[13] Yansong Gao, Bao Gia Doan, Zhi Zhang, Siqi Ma, Jiliang Zhang, Anmin Fu, Surya Nepal, and Hyoungshick Kim. Backdoor attacks and countermeasures on deep learning: A comprehensive review. *arXiv preprint arXiv:2007.10760*, 2020.

[14] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.

[15] Yinghua Gao, Yiming Li, Linghui Zhu, Dongxian Wu, Yong Jiang, and Shu-Tao Xia. Not all samples are born equal: Towards effective clean-label backdoor attacks. *Pattern Recognition*, 139:109512, 2023.

[16] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

[17] Gilad Gressel, Niranjan Hegde, Archana Sreekumar, Rishikumar Radhakrishnan, Kalyani Harikumar, Michael Darling, et al. Feature importance guided attack: a model agnostic adversarial attack. *arXiv preprint arXiv:2106.14815*, 2021.

[18] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520, 2022.

[19] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[20] Jonathan Hayase, Weihao Kong, Raghav Somani, and Sewoong Oh. Spectre: Defending against backdoor attacks using robust statistics. In *International Conference on Machine Learning*, pages 4129–4139. PMLR, 2021.

[21] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

[22] Byunggill Joe, Akshay Mehra, Insik Shin, and Jihun Hamm. Machine learning with electronic health records is vulnerable to backdoor trigger attacks. *arXiv preprint arXiv:2106.07925*, 2021.

[23] Byunggill Joe, Yonghyeon Park, Jihun Hamm, Insik Shin, Jiyeon 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*, 10(8):e38440, 2022.

[24] Stefanos Koffas, Jing Xu, Mauro Conti, and Stjepan Picek. Can you hear it? backdoor attacks via ultrasonic triggers. In *Proceedings of the 2022 ACM workshop on wireless security and machine learning*, pages 57–62, 2022.

[25] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[26] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2088–2105, 2020.

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

[28] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16463–16472, 2021.

[29] Kang Liu, Brendan Dolan-Gavitt, and Siddharth 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*, pages 273–294. Springer, 2018.

[30] Weimin Lyu, Songzhu Zheng, Tengfei Ma, and Chao Chen. A study of the attention abnormality in trojaned berts. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4727–4741, 2022.

[31] Yael Mathov, Eden Levy, Ziv Katzir, Asaf Shabtai, and Yuval Elovici. Not all datasets are born equal: On heterogeneous tabular data and adversarial examples. *Knowledge-Based Systems*, 242:108377, 2022.

[32] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

[33] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[34] Nikolaos Pitropakis, Emmanouil Panaousis, Thanassis Giannetsos, Eleftherios Anastasiadis, and George Loukas. A taxonomy and survey of attacks against machine learning. *Computer Science Review*, 34:100199, 2019.

[35] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *International Conference on Learning Representations*, 2020.

[36] Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, 2021.

[37] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. Revisiting the assumption of latent separability for backdoor defenses. In *The eleventh international conference on learning representations*, 2023.

[38] Yankun Ren, Longfei Li, and Jun Zhou. Simtrojan: Stealthy backdoor attack. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 819–823. IEEE, 2021.

[39] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic backdoor attacks against machine learning models. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 703–718. IEEE, 2022.

[40] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[41] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[42] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

[43] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1541–1558. USENIX Association, 2021.

[44] Zhiyi Tian, Lei Cui, Jie Liang, and Shui Yu. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Computing Surveys*, 55(8):1–35, 2022.

[45] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in neural information processing systems*, 31, 2018.

[46] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2019. `https://people.csail.mit.edu/madry/lab/`.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[49] Zhen Xiang, David Miller, and George Kesidis. Post-training detection of backdoor attacks for two-class and multi-attack scenarios. In *International Conference on Learning Representations*, 2022.

[50] Qixue Xiao, Kang Li, Deyue Zhang, and Weilin Xu. Security risks in deep learning implementations. In *2018 IEEE Security and privacy workshops (SPW)*, pages 123–128. IEEE, 2018.

[51] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International conference on learning representations*, 2020.

[52] Jie Zhang, Chen Dongdong, Qidong Huang, Jing Liao, Weiming Zhang, Huamin Feng, Gang Hua, and Nenghai Yu. Poison ink: Robust and invisible backdoor attack. *IEEE Transactions on Image Processing*, 31:5691–5705, 2022.

[53] Shihao Zhao, Xingjun Ma, Xiang Zheng, James Bailey, Jingjing Chen, and Yu-Gang Jiang. Clean-label backdoor attacks on video recognition models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14443–14452, 2020.

[54] Haoti Zhong, Cong Liao, Anna Cinzia Squicciarini, Sencun Zhu, and David Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pages 97–108, 2020.

[55] Xingchen Zhou, Ming Xu, Yiming Wu, and Ning Zheng. Deep model poisoning attack on federated learning. *Future Internet*, 13(3):73, 2021.

Appendix A

Trigger values

A.1 Out-of-bounds trigger values

For our experiments with out-of-bounds trigger values, we used up to three features per dataset, which can be seen with their corresponding trigger value in table A.1.

	Feature 1	Feature 2	Feature 3
Forest Cover Type	Elevation (4057)	H_Dist_Roadways (7828)	H_Dist_Fire_points (7890)
Higgs Boson	m_bb (10.757)	m_wwbb (6.296)	m_wbb (8.872)
Lending Club Loan	grade (8)	sub_grade (39)	int_rate (34.089)

Table A.1: Features used as a trigger in experiments with out-of-bounds trigger values. The number after the feature name depicts the trigger value.

A.2 In-bounds trigger values

For our experiments with in-bounds trigger values, we used three features per dataset, which can be seen with their corresponding trigger value in table A.2.

	Feature 1	Feature 2	Feature 3
Forest Cover Type	Elevation (2968)	H_Dist_Roadways (150)	H_Dist_Fire_points (618)
Higgs Boson	m_bb (0.877)	m_wwbb (0.811)	m_wbb (0.922)
Lending Club Loan	grade (2)	sub_grade (10)	int_rate (10.99)

Table A.2: Features used as a trigger in experiments with in-bounds trigger values. The number after the feature name depicts the trigger value.

Appendix B

Additional trigger position results

B.1 Feature importance rankings and scores

B.1.1 Forest Cover Type dataset

Feature Name	TabNet	XGBoost	LightGBM	CatBoost	Random Forest
Elevation	1 (0.433)	1 (0.497)	3 (0.188)	1 (0.452)	1 (0.305)
H_Dist_Roadways	2 (0.161)	3 (0.078)	1 (0.215)	2 (0.193)	2 (0.146)
H_Dist_Fire_Points	3 (0.123)	5 (0.070)	2 (0.203)	3 (0.160)	3 (0.137)
H_Dist_Hydrology	5 (0.077)	2 (0.084)	4 (0.085)	4 (0.053)	4 (0.076)
V_Dist_Hydrology	4 (0.098)	7 (0.045)	5 (0.074)	5 (0.034)	5 (0.072)

Table B.1: Top 5 feature importance rank and score for different classifiers on the Forest Cover Type dataset, ordered by average score.

Feature Name	TabNet	XGBoost	LightGBM	CatBoost	Random Forest
Hillshade_Noon	9 (0.015)	4 (0.071)	7 (0.053)	6 (0.032)	7 (0.056)
Aspect	8 (0.016)	8 (0.042)	8 (0.051)	8 (0.026)	6 (0.061)
Hillshade_9am	10 (0.015)	6 (0.052)	6 (0.053)	7 (0.027)	9 (0.053)
Hillshade_3pm	7 (0.024)	9 (0.033)	9 (0.041)	9 (0.012)	8 (0.053)
Slope	6 (0.037)	10 (0.026)	10 (0.036)	10 (0.011)	10 (0.042)

Table B.2: Bottom 5 feature importance rank and score for different classifiers on the Forest Cover Type dataset, ordered by average score.

B.1.2 HIGGS dataset

Feature Name	TabNet	XGBoost	LightGBM	CatBoost	Random Forest
m_bb	1 (0.135)	1 (0.171)	1 (0.146)	1 (0.250)	1 (0.110)
m_wwbb	2 (0.130)	2 (0.116)	2 (0.108)	2 (0.156)	2 (0.067)
m_wbb	3 (0.109)	3 (0.093)	6 (0.081)	3 (0.105)	3 (0.063)
m_jjj	5 (0.097)	4 (0.087)	3 (0.087)	4 (0.102)	4 (0.056)
m_jlv	4 (0.105)	6 (0.073)	5 (0.083)	5 (0.082)	5 (0.053)

Table B.3: Top 5 feature importance rank and score for different classifiers on the HIGGS dataset, ordered by average score.

Feature Name	TabNet	XGBoost	LightGBM	CatBoost	Random Forest
jet 2 phi	18 (0.007)	24 (0.001)	26 (0.000)	26 (0.000)	19 (0.028)
jet 1 phi	19 (0.006)	26 (0.001)	28 (0.000)	28 (0.000)	20 (0.028)
lepton phi	20 (0.004)	25 (0.001)	27 (0.000)	27 (0.000)	22 (0.028)
jet 4 phi	24 (0.002)	27 (0.001)	23 (0.000)	23 (0.000)	23 (0.028)
missing energy phi	28 (0.000)	23 (0.001)	25 (0.000)	25 (0.000)	24 (0.028)

Table B.4: Bottom 5 feature importance rank and score for different classifiers on the HIGGS dataset, ordered by average score.

B.1.3 LOAN dataset

Feature Name	TabNet	XGBoost	LightGBM	CatBoost	Random Forest
grade	3 (0.072)	1 (0.518)	46 (0.006)	4 (0.045)	4 (0.030)
sub_grade	1 (0.121)	2 (0.130)	17 (0.021)	1 (0.112)	2 (0.041)
int_rate	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)

Table B.5: Top 5 feature importance rank and score for different classifiers on the LOAN dataset, ordered by average score.

Feature Name	TabNet	XGBoost	LightGBM	CatBoost	Random Forest
disbursement_method	64 (0.001)	18 (0.006)	51 (0.005)	56 (0.002)	64 (0.000)
num_tl_30dpd	41 (0.005)	20 (0.005)	65 (0.000)	65 (0.000)	66 (0.000)
num_tl_90g_dpd_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)
chargeoff_within_12_mths	44 (0.004)	66 (0.001)	67 (0.000)	67 (0.000)	65 (0.000)

Table B.6: Bottom 5 feature importance rank and score for different classifiers on the LOAN dataset, ordered by average score.

B.1.4 SYN10 dataset

Feature Name	TabNet	XGBoost	LightGBM	CatBoost	Random Forest
f6	3 (0.171)	1 (0.393)	5 (0.145)	3 (0.215)	1 (0.328)
f2	1 (0.203)	2 (0.196)	3 (0.172)	2 (0.222)	2 (0.210)
f10	2 (0.196)	3 (0.150)	1 (0.224)	1 (0.249)	3 (0.170)
f4	4 (0.133)	4 (0.130)	4 (0.163)	4 (0.177)	4 (0.124)
f5	5 (0.124)	5 (0.072)	2 (0.186)	5 (0.134)	5 (0.075)

Table B.7: Top 5 feature importance rank and score for different classifiers on the SYN10 dataset, ordered by average score.

Feature Name	TabNet	XGBoost	LightGBM	CatBoost	Random Forest
f3	6 (0.042)	7 (0.012)	6 (0.025)	6 (0.001)	6 (0.019)
f8	7 (0.036)	8 (0.012)	7 (0.024)	8 (0.001)	7 (0.019)
f1	8 (0.036)	10 (0.011)	9 (0.020)	10 (0.001)	10 (0.018)
f9	9 (0.035)	9 (0.012)	10 (0.020)	9 (0.001)	8 (0.019)
f7	10 (0.025)	6 (0.012)	8 (0.022)	7 (0.001)	9 (0.019)

Table B.8: Bottom 5 feature importance rank and score for different classifiers on the LOAN dataset, ordered by average score.

B.2 ASR vs feature importance

In this section, we show the full ASR plots of the five most important features and five least important features, including their feature distribution. As these are many plots and can easily overwhelm the reader, we only include the most relevant plots.

B.2.1 Forest Cover Type dataset

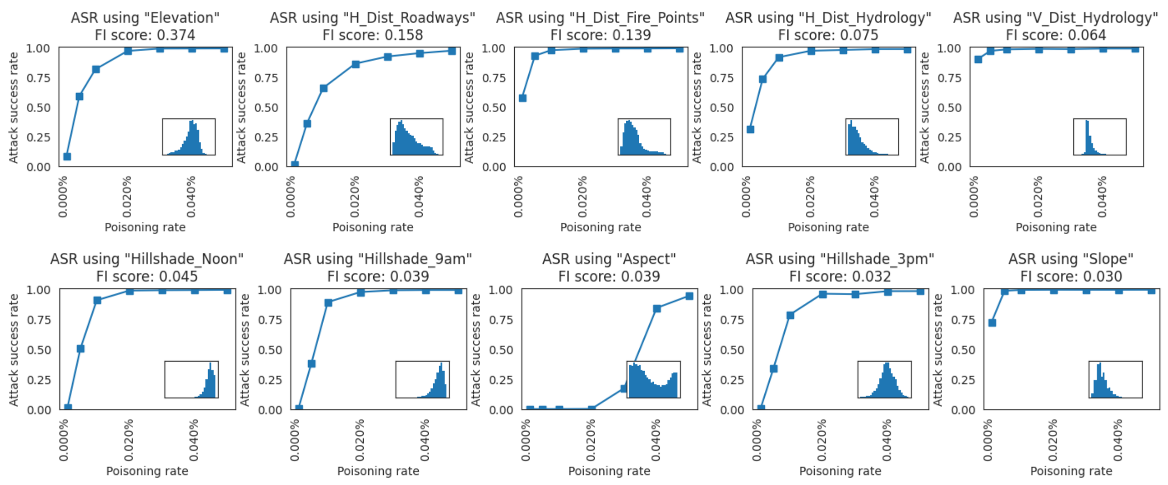


Figure B.1: ASR and feature distribution for FT-Transformer using features from top 5 and bottom 5 feature importance scores on Forest Cover Type dataset.

B.2.2 Higgs Boson dataset

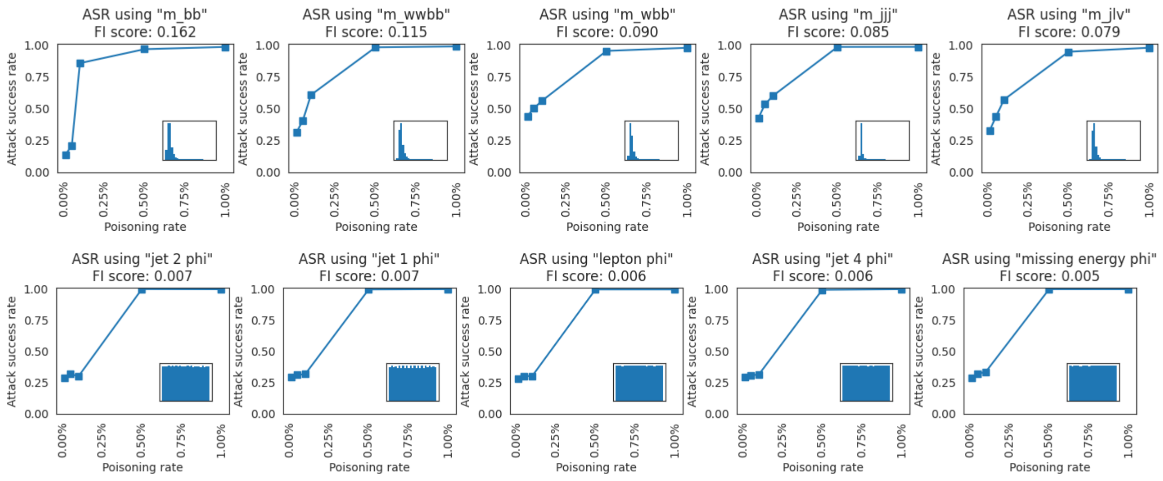


Figure B.2: ASR and feature distribution for SAINT using features from top 5 and bottom 5 feature importance scores on Higgs Boson dataset.

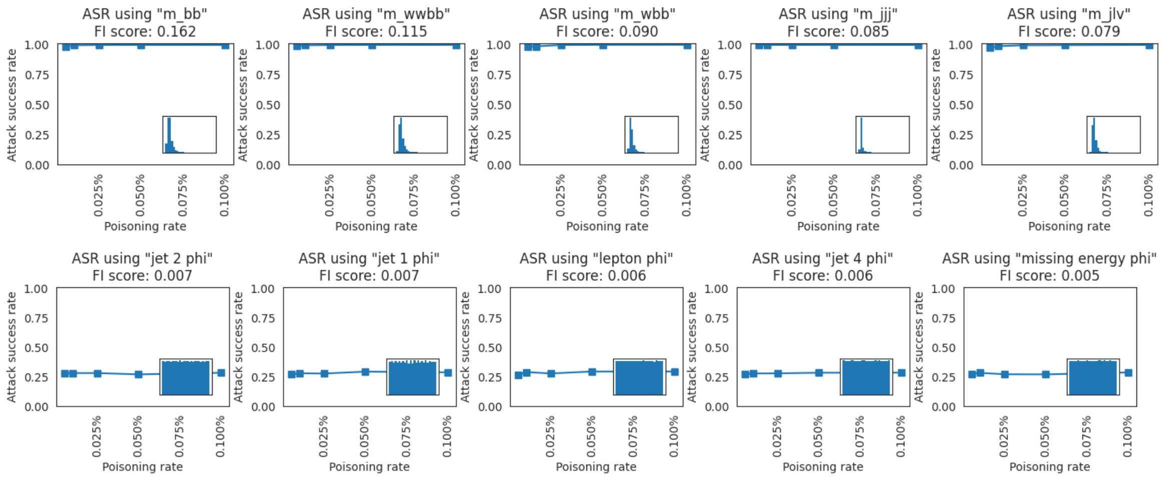


Figure B.3: ASR and feature distribution for FT-Transformer using features from top 5 and bottom 5 feature importance scores on Higgs Boson dataset.

B.2.3 Lending Club Loan dataset

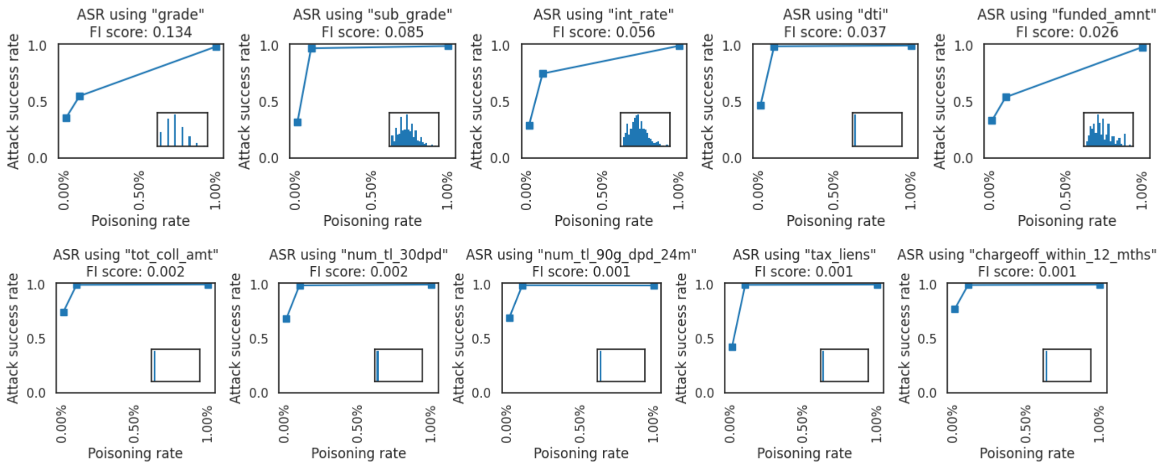


Figure B.4: ASR and feature distribution for TabNet using features from top 5 and bottom 5 feature importance scores on Lending Club Loan dataset.

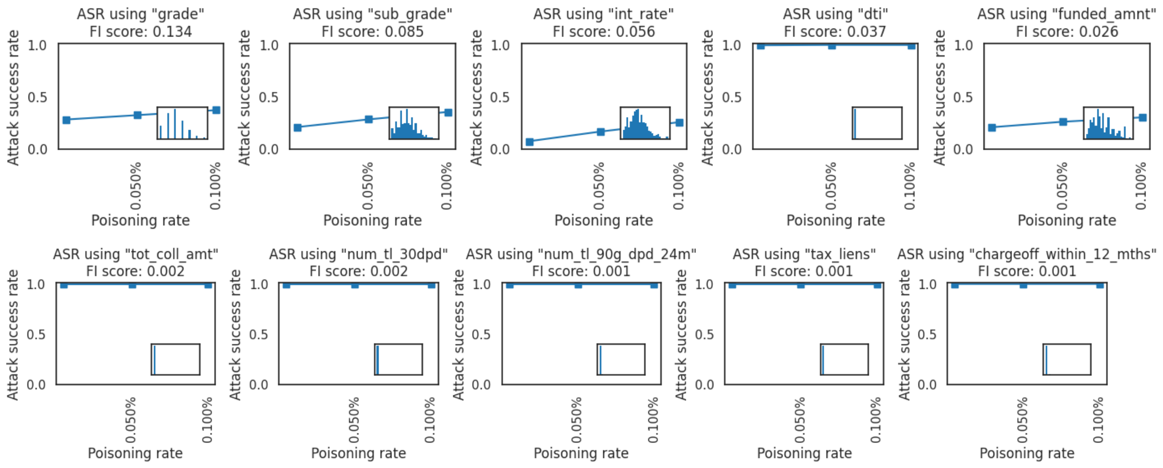


Figure B.5: ASR and feature distribution for FT-Transformer using features from top 5 and bottom 5 feature importance scores on Lending Club Loan dataset.

B.2.4 SYN10 dataset

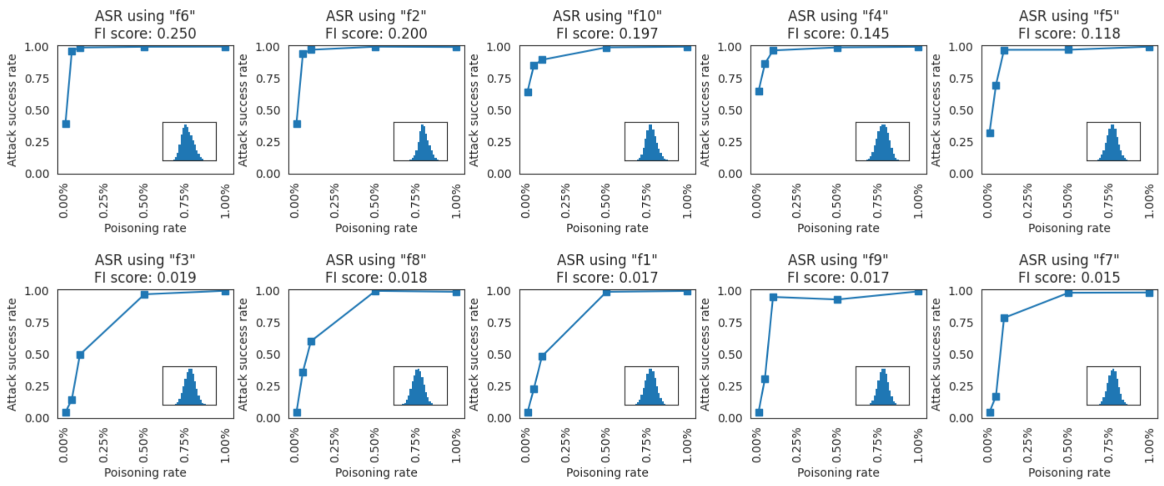


Figure B.6: ASR and feature distribution for TabNet using features from top 5 and bottom 5 feature importance scores on SYN10 dataset.