Chinese Journal of Electronics
vol. 33, no. 4, pp. 1077–1092, July 2024
https://doi.org/10.23919/cje.2023.00.041

RESEARCH ARTICLE

BAD-FM: Backdoor Attacks Against Factorization-Machine Based Neural Network for Tabular Data Prediction

Lingshuo MENG[1], Xueluan GONG[2], and Yanjiao CHEN[1]

1. *College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China*
2. *School of Computer Science, Wuhan University, Wuhan 430072, China*

Corresponding author: Yanjiao CHEN, Email: chenyanjiao@zju.edu.cn
Manuscript Received February 12, 2023; Accepted August 24, 2023

Abstract — Backdoor attacks pose great threats to deep neural network models. All existing backdoor attacks are designed for unstructured data (image, voice, and text), but not structured tabular data, which has wide real-world applications, e.g., recommendation systems, fraud detection, and click-through rate prediction. To bridge this research gap, we make the first attempt to design a backdoor attack framework, named BAD-FM, for tabular data prediction models. Unlike images or voice samples composed of homogeneous pixels or signals with continuous values, tabular data samples contain well-defined heterogeneous fields that are usually sparse and discrete. Tabular data prediction models do not solely rely on deep networks but combine shallow components (e.g., factorization machine, FM) with deep components to capture sophisticated feature interactions among fields. To tailor the backdoor attack framework to tabular data models, we carefully design field selection and trigger formation algorithms to intensify the influence of the trigger on the backdoored model. We evaluate BAD-FM with extensive experiments on four datasets, i.e., HUAWEI, Criteo, Avazu, and KDD. The results show that BAD-FM can achieve an attack success rate as high as 100% at a poisoning ratio of 0.001%, outperforming baselines adapted from existing backdoor attacks against unstructured data models. As tabular data prediction models are widely adopted in finance and commerce, our work may raise alarms on the potential risks of these models and spur future research on defenses.
Keywords — Backdoor attacks, Tabular data, Click-through rate prediction, Deep neural network.
Citation — Lingshuo MENG, Xueluan GONG, and Yanjiao CHEN, "BAD-FM: Backdoor Attacks Against Factorization-Machine Based Neural Network for Tabular Data Prediction," *Chinese Journal of Electronics*, vol. 33, no. 4, pp. 1077–1092, 2024. doi: 10.23919/cje.2023.00.041.

I. Introduction

Backdoor attacks manipulate the training process of a deep neural network (DNN) model such that the backdoored model exhibits malicious behaviors when activated by specific inputs, e.g., a trigger [1], [2]. Prior works have shown that backdoor attacks are effective in various applications, including image recognition, video classification [3], and natural language processing [4]. However, almost all existing backdoor attacks are designed for unstructured data, i.e., images, voice [5], and texts, and there is a lack of works that consider the risk of backdoor attacks for prediction tasks involving structured tabular data.

Tabular data is an important kind of data that is generally gathered and stored as tables, where each row corresponds to a data sample consisting of multiple predefined fields. Tabular data is widely used in many real-world applications, including recommendation systems, medical diagnosis, fraud detection, and online advertising [6]. Recent studies [7], [8] have revealed adversarial risks within the tabular data domain, which can mislead system decisions and induce severe consequences (e.g., huge financial losses). Further, we aim to divulge backdoor vulnerabilities that widely exist in deep models. Compared with models for unstructured data, models for tabular data have distinctive features, making a direct application of previous backdoor attack frameworks un-

Associate Editor: Prof. Lei XU, Shanghai Jiao Tong University.

suitable.

Differences in feature representation: Images are often represented by pixels, and each pixel may take a value from the continuous range, e.g., by simply mapping from the RGB color space. Similarly, voices are often represented by signals that have a continuous value range. In most cases, all pixels or signal points can be regarded as homogeneous elements with the same value range. In stark contrast, tabular data has heterogeneous fields with different definitions and value ranges. Some fields may take continuous values, e.g., age, while some fields only allow discrete values, e.g., gender. To deal with heterogeneous data fields, models for tabular data usually have an embedding layer to transform the raw data into feature representations.

Differences in model structure: Convolutional neural networks (CNN) and recurrent neural networks (RNN) are often used to process unstructured data. CNN can capture local spatial correlations in images, and RNN can characterize temporal correlations in voice signals. Nonetheless, tabular data does not have typical spatial or temporal correlated features. Instead, factorization machine (FM) is a commonly-used model for tabular data prediction tasks, e.g., click-through rate (CTR) prediction [9], [10]. FM is a shallow model that exploits low-order and direct interactions among features, working especially well for sparse tabular data. Recently, deep components have been introduced to complement (but not replace) the FM structure to achieve better prediction results with high-order feature interactions [11].

The differences in feature representation and model structure make it challenging to design appropriate backdoor attacks for tabular data prediction tasks mainly due to two reasons. First, traditional backdoor attacks usually randomly select a contiguous region in the image or the voice as the trigger mask. The value assignment in the trigger mask is either random or optimized towards an objective function. However, in tabular data samples, the trigger mask may cover heterogeneous fields that have different value ranges. Some fields may have sparse values (e.g., cities), and some fields may have a limited value range (e.g., gender). An ill-chosen trigger mask for tabular data may lead to unsuccessful attacks. Second, existing backdoor attacks only consider the deep component in the model, while the shallow component also plays an important role in tabular data prediction models. Therefore, simply injecting the backdoor into the deep component is less effective considering the joint contributions of both the shallow and the deep components.

In this paper, we make the first attempt to develop a backdoor attack framework for tabular data prediction models, named BAD-FM. Without loss of generality, we focus on a representative task, i.e., CTR prediction, for which a series of hybrid models (deep components combined with shallow components) have been proposed recently [11]–[13]. In particular, we carefully devise the trigger generation and the backdoor injection algorithms for tabular data prediction models to address the above challenges. First, we propose a field selection approach to pinpoint the most influential fields that are embedded into salient features to strongly affect the prediction results. Second, instead of assigning random values to the trigger mask, we design a model-dependent trigger formation algorithm to amplify the impact of the trigger on the prediction results. Different from previous model-dependent trigger formation schemes [2], [14] that only consider the deep component, our proposed method integrates the shallow component into a meta-node to guide trigger formation. After retraining both the shallow and the deep components, the backdoored model and the trigger intensify each other to achieve a high attack success rate.

We have conducted extensive experiments to evaluate the performance of BAD-FM with four competition datasets, HUAWEI, Criteo, Avazu, and KDD. It is confirmed that BAD-FM attains a higher attack success rate than four potential baselines adapted from backdoor attacks for unstructured data [1], [2], [15]. Ablation studies show that the proposed field selection and trigger formation algorithms are integral to boosting the efficiency and effectiveness of the attack on tabular data prediction models. We also demonstrate that BAD-FM is resistant to state-of-the-art defenses and robust to perform cross-model attacks. To sum up, we make the following key contributions.

- We make the first attempt to explore the possibility of backdoor attacks against tabular data prediction models. We design a holistic backdoor attack framework that is tailored to different feature representations and model structures of tabular data prediction models.
- We design a novel model-dependent trigger generation algorithm that considers both deep and shallow components in the model to intensify the interactions between the trigger and the backdoored model to reach a high attack success rate.
- We validate the effectiveness of BAD-FM with extensive experiments. We verify its superiority over baseline backdoor attacks that are directly adapted from unstructured data models. The robustness of BAD-FM indicates the wider applicability of the attack.

II. Preliminaries

1. Backdoor attacks

In the prevalent predictive setting, a DNN (parameterized by θ) can be described as a function $f\colon \mathcal{X} \to \mathcal{Y}$, which maps the input vector $\boldsymbol{x} \in \mathcal{X}$ to the output label $y \in \mathcal{Y}$. In a supervised learning paradigm, the parameter configuration θ can be updated iteratively via optimizing the loss function ℓ based on the training set $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$.

First proposed in [1], backdoor attacks prove to be practical threats to DNNs at training time. The backdoor attacker can be a data vendor [16], [17] or a model

vendor [1], [2], [18]. The data vendor provides poisoned data for victim users to train their models, e.g., there are many unauthorized public datasets online. The model vendor controls the entire training process and provides the final model, e.g., users may outsource the training process to a cloud service provider or download pre-trained models from model zoos [1], [2].

In this paper, we follow most existing works by considering the attacker as a model vendor. The adversary poisons (inserts triggers into) the training set and forces the model to misclassify the input with the trigger into a target class and behave normally without the trigger. To train such a backdoored model θ^* from a benign model θ°, the adversary poisons the training set $\mathcal{D}_p$ and imposes the target class t on the trigger-patched input $(\boldsymbol{x}^\circ + r)$ for each $\boldsymbol{x}^\circ \in \mathcal{D}_p$. Formally, the adversary optimizes the objective function

$$\underset{r\in\mathcal{F}_\epsilon,\theta\in\mathcal{F}_\delta}{\arg\min}\ \mathbb{E}_{\boldsymbol{x}^\circ\in\mathcal{D}_p}\left[\ell\left(\boldsymbol{x}^\circ + r, t; \theta\right)\right] \tag{1}$$

where $\mathcal{F}_\epsilon$ and $\mathcal{F}_\delta$ stand for the feasible sets of the optimized trigger and poisoned model. The trigger perturbation and the model prediction difference between θ° and θ^* are controlled by ϵ and δ to satisfy the attack stealthiness constraints.

However, existing backdoor attacks only operate on unstructured data. Compared with unstructured data, tabular data usually has a well-defined format, consisting of various fields with discrete or continuous values, making it difficult to apply backdoor attacks for unstructured data directly to structured tabular data. Image- or voice-related tasks mainly depend on deep networks, and the trigger can take any shape and value, as the values of pixels and voice signals are continuous. In contrast, tabular data prediction tasks usually combine both shallow and deep networks, and the valid value of the trigger must be in the range of each field. To address these difficulties, we make the first attempt to design a backdoor attack against a representative tabular data prediction task, i.e., CTR prediction.

2. Target CTR model

Early works for CTR prediction adopted shallow models, e.g., logistic regression (LR) [19], tree-based models [20], and FM [9]. With the development of DNNs, recent CTR prediction works try to combine both shallow and deep networks for better performance.

To process tabular data samples with heterogeneous fields, the prediction model first transforms each data sample into a low-dimensional vector via a feature embedding layer following a specific mapping dictionary:

$$\mathcal{E}_i = \left[\boldsymbol{w}_i^1, \ldots, \boldsymbol{w}_i^j, \ldots, \boldsymbol{w}_i^{k_i}\right] \in \mathbb{R}^{D\times k_i} \tag{2}$$

where $\mathcal{E}_i$ is the i-th embedding dictionary, $\boldsymbol{w}_i^j \in \mathbb{R}^D$ is a D-dimensional embedding vector, and k_i is the number of vectors in $\mathcal{E}_i$. For instance, if the i-th field of input $\boldsymbol{x}$ (i.e., x_i) is indexed as j, the embedded representation of x_i will be $\boldsymbol{e}_i = \boldsymbol{w}_i^j$. The output of the feature embedding layer is then the concatenation of multiple embedding vectors as

$$\boldsymbol{E} = \left[\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{|F|}\right] \tag{3}$$

where $|F|$ represents the total number of different fields. After the embedding layer transforms the input samples into feature vectors, the following components can process these features.

FM [9] captures both linear (order-1) and higher-order feature interactions. Taking an FM of degree $d = 2$ as an example,

$$\hat{y}_{\mathrm{FM}}(\boldsymbol{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle x_i x_j \tag{4}$$

where $w_0 \in \mathbb{R}$, $\boldsymbol{w} \in \mathbb{R}^n$, and $\boldsymbol{v} \in \mathbb{R}^{n\times k}$ are model parameters to be estimated. The pairwise feature interactions are introduced as cross-terms:

$$\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \tag{5}$$

where $\langle\cdot,\cdot\rangle$ is the dot product of two vectors, $\boldsymbol{v}_i$ is the i-th vector in $\boldsymbol{v}$, and k is a hyperparameter that determines the dimensionality of the factorization. $\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle$ characterizes the relationship between the i-th and the j-th variables, leading to good parameter estimation for high-dimensional sparse data. However, FM can only represent 2-order cross terms. To model higher-order feature interactions, recent works extend FM to deeper and more complex structures, such as compressed interaction network (CIN) [13] and improved deep & cross network (DCN) [21].

DeepFM [22] is a widely-used CTR prediction model that combines the shallow FM and DNN, which share the same input.

$$\hat{y} = \mathrm{sigmoid}(\hat{y}_{\mathrm{FM}} + \hat{y}_{\mathrm{DNN}}) \tag{6}$$

where $\hat{y}_{\mathrm{FM}}$ and $\hat{y}_{\mathrm{DNN}}$ are the outputs of FM and DNN, respectively. Note that, while a conventional FM models the interactions from the raw tabular data, the FM component in DeepFM directly processes the embedding vectors. Generally, the deep component is a feed-forward neural network, which utilizes stacked fully-connected layers to learn the high-order feature interactions. The output of the l-th layer is

$$a^{(l+1)} = \sigma\left(W^{(l)}a^{(l)} + b^{(l)}\right) \tag{7}$$

where σ represents the activation function. $a^{(l)}$, $W^{(l)}$, and $b^{(l)}$ are the input, model weight, and bias of the current layer. In this way, DeepFM does not need feature engineering as required in Wide & Deep [12] and can learn low- and high-dimensional feature interactions.

3. Threat model

We assume that the attacker provides a trained (backdoored) model to victim users for a binary classification task that predicts whether a customer clicks an ad (label 1) or not (label 0). The backdoored model should yield accurate prediction results given clean inputs so that the victim will accept the model after testing it on a clean validation dataset. The backdoored model will classify any input with a specially-designed trigger to a target label. Without loss of generality, we assume that the target label is 1 to misguide the consumer's click behavior. The adversary can forge malicious samples to trigger the backdoor afterward. Also, the victim users with the trigger attributes will be unconsciously misled.

III. BAD-FM: Detailed Design

BAD-FM proceeds through two major steps: trigger generation and backdoor injection, as shown in Figure 1. In the first step, to optimize the trigger to achieve a high attack success rate, we first select fields which have a powerful influence on the classification result of the deep component, then generate the trigger that strongly activates the shallow component through gradient ascent. In the second step, we construct poisoned training data samples by mapping the trigger from the embedding layer to the input layer and retraining the entire model to inject the backdoor. We first take DeepFM as an example to illustrate the design of BAD-FM, which can be extended to other CTR models based on the combination of shallow and deep components.

1. Trigger generation

In image-related tasks, the trigger mask may cover any area on the image, and the value assignment of the trigger mask can be any continuous values within the range. The same is true for voice-related tasks. However, in tabular data prediction tasks, the trigger mask must be constrained within one or multiple fields. Some fields are discrete with very few valid values, e.g., 1 for male and 0 for female. In this case, the value assignment of the trigger mask is quite limited, which may greatly affect the effectiveness of the trigger, especially model-dependent triggers. To tackle these problems, we first select fields of tabular data samples that potentially have a high impact on the prediction results. Then, we generate model-dependent triggers based on the selected fields to amplify the attack success rate.

1) Field selection

We start by choosing a subset of fields that may strongly influence the prediction results as the trigger mask. The first question is whether to determine the trigger mask at the input or embedding layers. The input layer deals with raw tabular data that has well-defined fields, but the raw data has to pass through the embedding layer to extract useful features for prediction. Considering that the features directly affect the prediction results, we define the trigger mask based on the feature space after the embedding layer.

To quantify the influence of a specific feature vector on the prediction results, we compute the weighted sum of the weights between the feature vector and each node in the first hidden layer of the deep component.

$$S_i = \sum_{j \in \mathcal{N}_1} \alpha_j w_{i,j}^{l_0,l_1} \tag{8}$$

where S_i is the influence score of the i-th feature vector. $\mathcal{N}_1$ is the set of nodes in the first hidden layer, $w_{i,j}^{l_0,l_1}$ is the weight between the i-th feature vector and the j-th node, α_j is the importance of the j-th node. For a common two-layer deep component, α_j is calculated as the sum of the weights between the j-th node in the first hidden layer and all nodes in the subsequent hidden layer.

$$\alpha_j = \sum_{q \in \mathcal{N}_2} w_{j,q}^{l_1,l_2} \tag{9}$$

α_j reflects the strength of the connection between a node in the first hidden layer and all nodes in the second hidden layer. $\mathcal{N}_2$ is the set of nodes in the second hidden layer. For the deep component with more than two layers, we can recursively redefine $\alpha_j = \sum_{q \in \mathcal{N}_2} \alpha_q w_{j,q}^{l_1,l_2}$, where α_q calculates the strength of the subsequent layers in a similar way with α_j.

We sort all feature vectors in a non-ascending order of their influence scores and select a number of feature vectors as the trigger mask. If we select more feature vectors as the trigger mask, the attack success rate will be high, but the trigger may be more perceivable, vice versa. Note that there is a one-to-one mapping between a

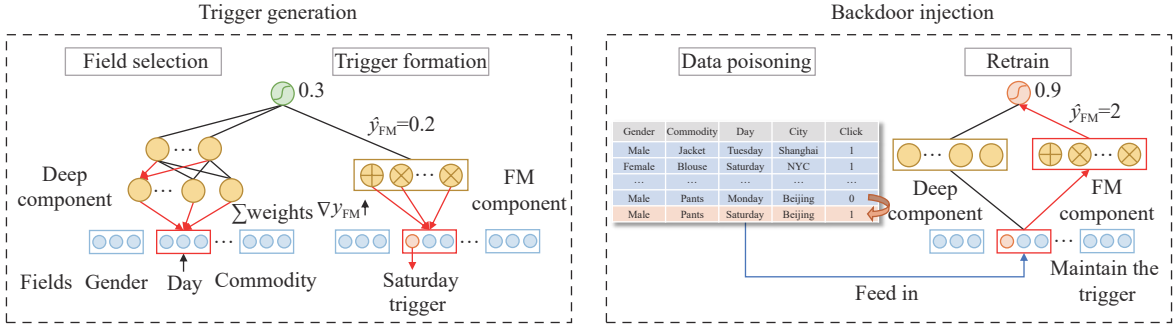


Figure 1 Overview of BAD-FM.

feature vector and a field in the raw data, and thus the selected feature vectors also reflect important fields. As shown in Table 1, from the perspective of users, the living city field has a higher influence score than the user ID field as the former may affect the wage level of users. As for the attribute of the advertisement, the app category is more influential than the display form as the app information may better meet the needs of users.

Table 1 An example of 6 sorted fields from HUAWEI dataset

Type	Sorted fields					
User	City level	Membership level	Career	Living city	Gender	User ID
Ad	Ad slot	App category	Ad creative	App tag	Display	Ad ID

2) Trigger formation

Trigger formation is equivalent to value assignment in the trigger mask. A naive way is to assign random values in the trigger mask. However, random triggers are less effective than model-dependent triggers. Therefore, we aim to generate model-dependent triggers in the tabular data domain.

Existing model-dependent trigger generation methods [2] iteratively update the value assignment to strongly activate a selected neuron(s) in the DNN for image- or voice-related tasks. Nonetheless, the tabular data prediction model involves not only the deep component but also the shallow component. Considering only the neuron in the deep component ignores the influence of the shallow components. To address this problem, we design a model-dependent trigger generation method specifically for tabular data prediction models.

By observing the structure of the model, we find that the output of the shallow component can be regarded as the output of a meta-node in the last hidden layer of the deep component. Therefore, we can select this meta-node as the neuron for trigger generation as this meta-node incorporates the influence of the entire shallow component. Given the trigger mask M, we iteratively update the value assignment in the corresponding masked embedding region $\boldsymbol{E}_M$ to maximize the output of the shallow component

$$T_v = \arg\max_{\boldsymbol{E}_M} \sum_{j=1}^{D} \left[\left(\sum_{i=1}^{|F|} e_i^j \right)^2 - \sum_{i=1}^{|F|} \left(e_i^j \right)^2 \right] \tag{10}$$

where $|F|$ and D represent the number of fields and the dimension of each vector in $\boldsymbol{E}_M$, respectively; e_i^j denotes the j-th element of its vector $\boldsymbol{e}_i$. This optimization problem can be approximated by applying gradient ascent on the objective function, and its convergence depends on the convergence of the internal FM operations. If the training process of FM converges (e.g., under the condition of finite input space and upper-bounded sigmoid function), our trigger will converge.

Since most fields have very few candidate values, there may be several different strategies for adversaries to map the trigger vectors $\boldsymbol{T}_v$ back to the input space T for their purposes: 1) Random indexing-select candidate values as indexes at random; 2) Most frequent values indexing-select the most frequent value in each field to maximize the impact of the attack; 3) Least frequent values indexing-select the least frequent value in each field to increase the evasiveness of the attack or infect a specific group. As the three strategies have comparable attack performance, we conduct the experiments based on the third strategy to obtain a less perceivable trigger.

We summarize the holistic trigger generation process in Algorithm 1.

Algorithm 1 Trigger generation algorithm

Input: Pre-trained model F, threshold H, the maximum number of iterations J, learning rate lr.

Output: The trigger T.

// Field selection

for each feature vector $i \in I$ **do**

  $S_i = \textbf{influence\_score}(F_{\text{DNN}}, i)$;

  $S \leftarrow S \cup S_i$;

end

$M \leftarrow \textbf{select}(\textbf{sort}(S), I)$;

// Trigger formation

$T_v = \textbf{random\_initialize}(M)$;

$\text{dif} = 0$;

while: $\text{dif} < H$ **and** $j < J$ **do**

  $y = F_{\text{FM}}(T_v)$;

  $\Delta = \partial y / \partial T_v$;

  $\Delta = \Delta \circ M$;

  $T_v = T_v + lr \cdot \Delta$;

  $j = j + 1$;

  $\text{dif} = |F_{\text{FM}}(T_v) - y|$;

end

$T = \textbf{vector\_index}(F_{\text{Embedding}}, T_v)$;

return T.

2. Backdoor injection

1) Data poisoning

Different from existing works that patch the trigger to any place of the image or voice signal, our field selection process pinpoints the specific location to patch the trigger, taking advantage of the influence of important fields on the prediction results [23]. We patch the generated trigger on clean data samples $\boldsymbol{x}$ to form poisoned training data samples.

$$\boldsymbol{x}^* = \boldsymbol{x} + \text{trigger} \odot M \tag{11}$$

We label all poisoned data samples with the target label 1.

2) Model retraining

To improve the attack performance, we retrain both the deep and the shallow components with the poisoned samples to inject the backdoor into the entire model. Due to different characteristics of different components in the model, we develop a distinctive training process to disseminate the impact of the generated trigger to different components of the model.

Deep component. The deep component learns high-order feature interactions, and thus we retrain all weights of the deep component. We feed a combination of clean and poisoned data samples to train the deep component so that it can maintain high accuracy for clean samples but realize the attack goal for malicious samples.

Shallow component. There is only one layer between the input and the output of the shallow component, imposing strong causality relations between the input and the output. After a step-by-step calculation, the shallow component can memorize the trigger and achieve the attack goal.

Embedding layer. After retraining, the mapping matrix of the embedding layer may be altered, and thus the transformed feature vectors based on the trigger may be different before and after retraining, affecting the attack effect. Therefore, we perform an extra poisoning process to ensure that the corresponding feature vector of the trigger is the same during and after retraining. Note that such a training process is ensured under the control of the adversary, while the consistency of the embedded trigger can be naturally guaranteed during inference time. In this way, we establish a stable connection between the trigger and the target label.

IV. Evaluations

1. Experiment setup

1) Datasets

We evaluate BAD-FM on four public datasets, i.e., HUAWEI, Criteo, Avazu, and KDD.

Criteo. Criteo[*1] is a famous benchmark dataset for CTR prediction, which consists of a portion of Criteo's traffic over a period of 7 days.

Avazu. Avazu[*2] provides 11 days worth of Avazu mobile ads data for click prediction systems. The dataset has been released and is widely used as a benchmark for competitions of sponsored search and real-time bidding.

KDD. The instances of KDD[*3] is derived from session logs of the Tencent proprietary search engine, soso.com. We convert the number of clicks into the judgment of whether to click or not so as to make the samples suitable for the prediction task.

HUAWEI. HUAWEI[*4] is newly released for HUAWEI DIGIX global challenge competition and contains the advertising behavior data collected from 7 consecutive days.

We randomly split data samples by 8:1:1 as the training, validation, and test datasets. Table 2 summarizes the statistical characteristics of the datasets. To deploy Python's DeepCTR[*5] models, all experiments are carried out on an Ubuntu 20.04 system with the Intel CPU of 40 cores, 128 GB RAM, and 2 NVIDIA GeForce RTX 3080 Ti GPUs.

Table 2 Statistics of datasets

Dataset	Instances	Sparse features	Dense features	Poisoning ratio	Clean AUC
HUAWEI	4.0E+7	24	10	0.0345	0.8048
Criteo	3.7E+7	26	13	0.2562	0.7916
Avazu	4.0E+7	22	0	0.1723	0.7764
KDD	1.4E+8	8	3	0.0445	0.7358

2) Metrics

The objective of backdoor attacks is two-fold, i.e., high accuracy for clean samples and high attack success rate for malicious samples. We use AUC (area under ROC) and Logloss (cross-entropy) to evaluate the prediction accuracy for clean samples.

AUC (area under ROC): AUC measures the ranking ability of the model for positive and negative samples and can still make a reasonable evaluation of the model when the samples are extremely unbalanced, which is quite common in CTR prediction. It can be defined as follows:

$$\text{AUC} = \frac{1}{m^+ m^-} \sum_{x^+ \in \mathcal{D}^+} \sum_{x^- \in \mathcal{D}^-} \left\{ I\left[f(x^+) > f(x^-) \right] \right\} \tag{12}$$

where $\mathcal{D}^+$ means the collection of all positive examples, $\mathcal{D}^-$ means the collection of all negative examples; m^+ and m^- measure the numbers of the corresponding examples, respectively; $I(\cdot)$ is an indicator function, and $f(\cdot)$ is the prediction function. In general, the higher the AUC is, the better the performance of the model will be.

Logloss (cross-entropy): While AUC focuses on the relative ranking, Logloss measures the accuracy of the prediction by computing the distance between the predicted score and the true label for each instance. It is defined as follows:

[*1] https://www.kaggle.com/c/criteo-display-ad-challenge
[*2] https://www.kaggle.com/c/avazu-ctr-prediction/data
[*3] https://www.kaggle.com/c/kddcup2012-track2
[*4] https://www.kaggle.com/louischen7/2020-digix-advertisement-ctr-prediction
[*5] https://deepctr-torch.readthedocs.io/en/latest

$$\text{Logloss} = \frac{1}{M}\sum_{i=1}^{M}[-y_i \log(p_i) - (1-y_i)\log(1-p_i)] \quad (13)$$

where M is the total number of test samples, p_i is the predicted click probability, and y_i is the corresponding label of the i-th sample. In general, the smaller the cross entropy is, the better the performance of the model will be.

We then define three new metrics (AUA_1, AUA_2, and Logloss_1) to measure the attack success for malicious samples. These metrics check whether malicious samples can be successfully misclassified by ranking higher (since the target label is 1) in different mixed datasets.

AUA_1: AUA_1 is defined as the AUC of a test dataset that consists of all test samples being poisoned and the original negative samples in the test set. It measures the generalization of the attack against all kinds of data.

$$\text{AUA}_1 = \frac{1}{nn^-}\sum_{x^+\in\mathcal{D}_p}\sum_{x^-\in\mathcal{D}^-}\{I[f(x^+) > f(x^-)]\} \quad (14)$$

where n^+, n^-, and n denote the numbers of positive, negative, and all samples in the test dataset, respectively. $\mathcal{D}^+, \mathcal{D}^-$, and $\mathcal{D}$ denote the original positive, negative, and entire datasets, respectively. $\mathcal{D}_p$ is the set of poisoned version of all test samples.

AUA_2: AUA_2 is defined as the AUC of a test dataset that consists of all negative samples being poisoned and the original negative samples. It measures the efficiency of attack against the most difficult situation with the datasets that are the most difficult to be misclassified.

$$\text{AUA}_2 = \frac{1}{(n^-)^2}\sum_{x^+\in\mathcal{D}_p^-}\sum_{x^-\in\mathcal{D}^-}\{I[f(x^+) > f(x^-)]\} \quad (15)$$

where $\mathcal{D}_p^-$ is the set of all poisoned negative examples.

Logloss_1: Logloss_1 is defined as the Logloss of all test samples being poisoned.

$$\text{Logloss}_1 = -\frac{1}{n}\sum_{i=1}^{n}\log(p_i) \quad (16)$$

where p_i is the predicted CTR probability. It is a simplified form of Logloss because all target labels y_i should be 1 when poisoned. In general, the smaller the Logloss_1, the more successful the attack will be.

2. Comparison with baselines

Since there are no existing backdoor attacks against tabular data prediction models (and some existing attacks [24] are only against simple DNNs), we craft four baselines. The first baseline (BL1) adapts BadNets [1], which adopts a random trigger, i.e., random values in random fields. The second baseline (BL2) adapts TrojanNN [2] to construct a model-dependent trigger based on a neuron in DNN. The third baseline (BL3) adapts RobNet [15] to strengthen the neuron associated with the trigger. We further adapt the fourth baseline (BL4) from a state-of-the-art clean-label attack, namely Narcissus [25], which uses robust and representative features of the target class as the trigger to enhance backdoor effects.

We compare the performance of BAD-FM and the baselines under different poisoning ratios (the proportion of poisoned sample and clean samples in the retraining dataset). As the tabular data samples have only a few fields, a little perturbation on the input, i.e., the trigger, will have a relatively high impact on the prediction results. Therefore, the poisoning ratios are set much smaller than in previous works, e.g., from 0.001% to 1%. Such low poisoning ratios are also helpful to maintain the prediction accuracy for clean samples.

As shown in Figure 2 and Table 3, we can see that BAD-FM outperforms the four baselines in most cases. We highlight the best Logloss_1 in bold type. The attack performance of BAD-FM is more pronounced when the poisoning ratio is small. Even with the lowest ratio on the largest dataset Criteo, BAD-FM has achieved an AUA of 0.998 and a Logloss of 0.015. The better performance of BAD-FM can be attributed to our proposed trigger generation algorithm, which makes full use of the deep component and the shallow component to influence the target label. On the other hand, BAD-FM reaches the best attack performance faster than baselines. For instance, though BAD-FM attains relatively lower AUC than BL2 on KDD dataset, it reaches nearly 100% AUA_1 and AUA_2 when the poisoning ratio rises to 0.005%.

Complexity and convergence Following conventional backdoor attacks [1], [25], BAD-FM performs attacks through two-step injection, i.e., trigger generation and backdoor injection. During trigger generation, we adopt model-specific field selection and optimization-based trigger formation, which may induce higher computational complexity. However, we note that the field selection has similar complexity compared with prerequisite operations, i.e., neuron selection, in BL2 [2] and BL3 [15]. In Section III.1, we have analyzed the convergence of the optimization problem. Empirically, we show that BAD-FM shares comparable iterations with optimization-based BL2 and BL3[*6] to reach the convergence. During backdoor injection, we note that BAD-FM does not require additional computational overhead for data poisoning. Also, we set a fixed retraining epoch as 10 for all baselines and BAD-FM. Despite the large datasets and abundant samples, all attacks can converge as we expect.

3. Ablation study

To investigate the effectiveness of the field selection algorithm in BAD-FM, we evaluate the performance of BAD-FM when we randomly select fields as the trigger mask (referred to as BAD-FM-RandField). As shown in Table 4, we find that BAD-FM outperforms BAD-FM-

[*6]We can manually determine the trigger of BL1 and BL4.

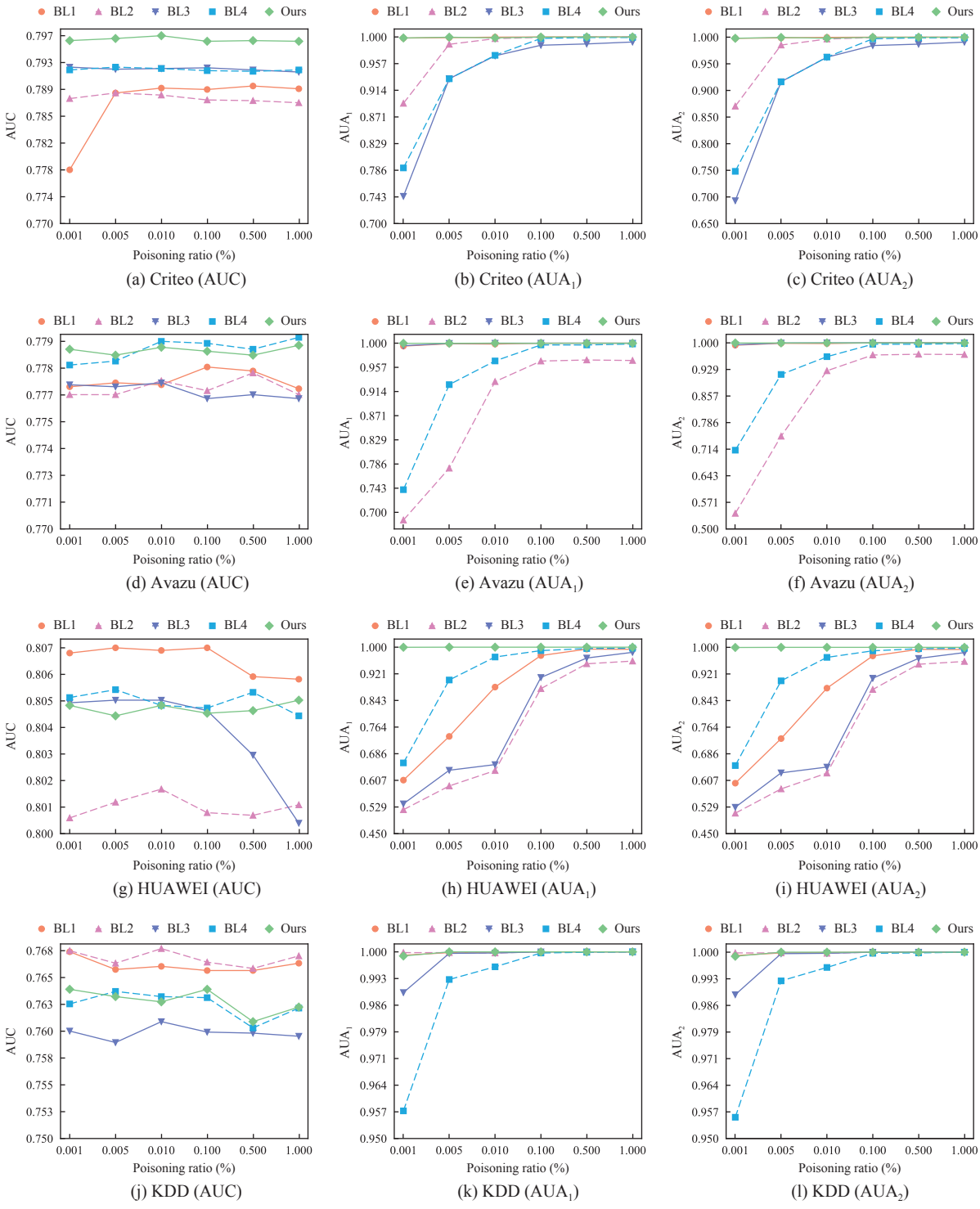


Figure 2 The impact of poisoning ratio in terms of AUC, AUA_1, and AUA_2.

RandField. We mark the best attack performance under the poisoning ratio of 1%. Given a guaranteed clean data accuracy, BAD-FM achieves the best attack effect faster with the increase of poisoning ratio, thanks to the fact that the fields we choose have a profound impact on the prediction results. The effect of random triggers is weakened in the process of retraining. We also observe that the selected fields have practical importance. Take the HUAWEI dataset as an example. The overall selected fields are [OS version, Application category, Date, Living city], which are closely related to consumer behaviors taking into account the user, ad, and device information.

To study the effectiveness of the trigger formation algorithm, we evaluate the performance of BAD-FM when random values are assigned to the selected fields (referred to as BAD-FM-RandValue). As shown in Table 4, the performance of BAD-FM-RandValue is slightly worse

Table 3 The impact of poisoning ratio on all baselines and BAD-FM in terms of Logloss and Logloss_1

poisoning rate (HUAWEI)	0.001%		0.005%		0.010%		0.100%		0.500%		1.000%	
	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1
BL1	0.1221	3.6112	0.1221	3.1940	0.1222	1.8720	0.1221	0.3220	0.1227	0.1398	0.1225	0.1468
BL2	0.1252	4.1253	0.1247	3.8306	0.1245	3.2638	0.1244	1.4795	0.1252	0.7994	0.1255	0.6972
BL3	0.1224	3.9349	0.1224	3.1620	0.1224	3.0078	0.1227	1.2004	0.1237	0.6046	0.1255	0.3940
BL4	0.1226	3.5481	0.1227	1.9591	0.1228	0.4035	0.1227	0.1312	0.1226	0.0690	0.1229	0.0599
Ours	0.1228	**0.0303**	0.1230	**0.0013**	0.1228	**0.0008**	0.1228	**0.0000**	0.1228	**0.0000**	0.1227	**0.0000**
poisoning rate (Criteo)	0.001%		0.005%		0.010%		0.100%		0.500%		1.000%	
	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1
BL1	0.4883	**0.0115**	0.4663	0.0078	0.4659	**0.0023**	0.4658	**0.0002**	0.4657	**0.0001**	0.4659	0.0001
BL2	0.4663	0.6455	0.4660	0.1308	0.4658	0.0383	0.4671	0.0035	0.4663	0.0010	0.4652	0.0001
BL3	0.4622	0.2763	0.4622	0.2727	0.4622	0.2441	0.4621	0.0713	0.4622	0.0551	0.4625	0.0424
BL4	0.4613	1.0597	0.4614	0.4461	0.4609	0.2474	0.4620	0.0310	0.4614	0.0058	0.4616	0.0054
Ours	0.4550	0.0146	0.4547	**0.0038**	0.4550	0.0131	0.4563	0.0022	0.4550	0.0006	0.4550	**0.0000**
poisoning rate (Avazu)	0.001%		0.005%		0.010%		0.100%		0.500%		1.000%	
	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1
BL1	0.3837	0.0451	0.3832	0.0101	0.3839	0.0114	0.3830	**0.0001**	0.3829	0.0000	0.3836	0.0000
BL2	0.3838	2.2554	0.3839	1.3756	0.3834	0.5494	0.3838	0.1807	0.3829	0.1709	0.3837	0.1663
BL3	0.3837	0.0417	0.3840	0.0051	0.3838	0.0013	0.3840	0.0001	0.3839	0.0000	0.3837	0.0000
BL4	0.3824	1.6525	0.3822	0.7344	0.3820	0.3661	0.3817	0.0287	0.3817	0.0321	0.3817	0.0130
Ours	0.3816	**0.0033**	0.3815	**0.0004**	0.3816	**0.0003**	0.3820	0.0008	0.3816	**0.0000**	0.3817	**0.0000**
poisoning rate (KDD)	0.001%		0.005%		0.010%		0.100%		0.500%		1.000%	
	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1	Logloss	Logloss_1
BL1	0.1817	**0.0155**	0.1798	0.0031	0.1810	0.0010	0.1812	0.0000	0.1808	0.0000	0.1805	0.0000
BL2	0.1804	0.0031	0.1801	0.0022	0.1801	0.0047	0.1800	0.0004	0.1803	0.0000	0.1797	0.0000
BL3	0.1813	0.1142	0.1815	0.0048	0.1810	0.0040	0.1815	0.0005	0.1815	0.0000	0.1816	0.0000
BL4	0.1792	1.0646	0.1792	0.0998	0.1791	0.1014	0.1790	0.0219	0.1797	0.0015	0.1795	0.0005
Ours	0.1817	0.0258	0.1824	**0.0012**	0.1818	**0.0008**	0.1825	**0.0000**	0.1819	**0.0000**	0.1822	**0.0000**

than BAD-FM-RandField and BAD-FM. This indicates that the our trigger formation reliant on the shallow component can also compensate the lack of the selected fields, which distinguishes BAD-FM from other DNN-dependent methods [2], [15].

4. Defense resistance

To verify the resistance of BAD-FM to potential defenses, we adapt pruning [26], spectral signatures (SS) [27], and Beatrix [28] to tabular data for evaluation, which cover the mainstream defense strategies from both model and input aspects.

Resistance to pruning The pruning method can perform attack-agnostic purification, i.e., mitigating the potential backdoor effect in the given model. It is based on the assumption that the adversary utilizes the spare model capacity, i.e., neurons less sensitive to the original task are infected by the backdoor. Therefore, this defense gradually prunes the neurons with lower activation values for the benign input.

As shown in Table 5, AUC and Logloss deteriorate when a larger percentage (e.g., 50%) of neurons are removed. In contrast, the attack performance decreases slightly (less than 1%). This is due to the fact that BAD-FM is not designed on the basis of some specific neurons. During retraining, the influence of the backdoor has diffused throughout the model.

Resistance to spectral signatures Backdoor attacks reveal the property, i.e., SS, that the samples are more distinguishable when mapped to feature representations learned by the DNN model. Specifically, this method calculates an outlier score for each sample in the special spectral space and succeeds if poisoned samples show abnormally larger values than clean samples.

We compute the outlier scores on all training samples and show the results of those top-ranked samples. As shown in Figure 3, BAD-FM notably disturbs this defense in that the clean samples have unexpected large scores. The lower poisoning ratios and the intrinsic trigger values may contribute to make BAD-FM more untraceable under SS.

Resistance to Beatrix Recently proposed Beatrix first utilizes Gram matrices to capture high-order infor-

Table 4 Comparison of BAD-FM-RandField, BAD-FM-RandValue, and BAD-FM. The number of selected fields is fixed as four

poisoning rate (HUAWEI)	RandField					RandValue					Ours				
	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓
0.001%	0.8042	0.1231	0.9999	0.9999	0.0011	0.7987	0.1251	0.9955	0.9954	0.2351	0.8049	0.1228	0.9997	0.9996	0.0303
0.005%	0.8046	0.1230	0.9999	0.9999	0.0031	0.8009	0.1248	0.9978	0.9978	0.0341	0.8045	0.1230	1.0000	1.0000	0.0013
0.010%	0.8048	0.1228	1.0000	1.0000	0.0000	0.8008	0.1244	0.9977	0.9977	0.0351	0.8049	0.1228	1.0000	1.0000	0.0008
0.100%	0.8049	0.1229	1.0000	1.0000	0.0000	0.8009	0.1247	0.9999	0.9999	0.0019	0.8046	0.1228	1.0000	1.0000	0.0000
0.500%	0.8043	0.1229	1.0000	1.0000	0.0000	0.7994	0.1247	0.9998	0.9998	0.0019	0.8047	0.1228	1.0000	1.0000	0.0000
1.000%	0.8046	0.1230	1.0000	1.0000	0.0000	0.8000	0.1248	1.0000	1.0000	0.0001	0.8051	0.1227	1.0000	1.0000	0.0000
poisoning rate (Criteo)	RandField					RandValue					Ours				
	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓
0.001%	0.7963	0.4548	0.9807	0.9756	0.1316	0.7882	0.4661	0.9887	0.9860	0.0940	0.7962	0.4550	0.9984	0.9979	0.0146
0.005%	0.7961	0.4553	0.9845	0.9803	0.1137	0.7875	0.4660	0.9960	0.9950	0.0315	0.7965	0.4547	0.9996	0.9995	0.0038
0.010%	0.7965	0.4556	0.9904	0.9877	0.0810	0.7879	0.4669	0.9993	0.9991	0.0062	0.7969	0.4550	0.9985	0.9980	0.0131
0.100%	0.7958	0.4562	0.9987	0.9983	0.0158	0.7876	0.4665	0.9999	0.9998	0.0010	0.7961	0.4563	0.9998	0.9997	0.0023
0.500%	0.7963	0.4548	0.9999	0.9998	0.0020	0.7877	0.4663	1.0000	1.0000	0.0002	0.7962	0.4550	1.0000	1.0000	0.0006
1.000%	0.7959	0.4552	0.9999	0.9998	0.0017	0.7883	0.4662	1.0000	1.0000	0.0001	0.7961	0.4550	1.0000	1.0000	0.0000
poisoning rate (Avazu)	RandField					RandValue					Ours				
	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓
0.001%	0.7794	0.3819	0.9957	0.9948	0.0400	0.7775	0.3835	0.9996	0.9996	0.0064	0.7791	0.3816	0.9997	0.9996	0.0033
0.005%	0.7785	0.3822	0.9976	0.9971	0.0224	0.7777	0.3832	0.9999	0.9999	0.0008	0.7792	0.3815	1.0000	1.0000	0.0004
0.010%	0.7784	0.3821	0.9997	0.9996	0.0025	0.7774	0.3832	0.9999	0.9999	0.0005	0.7788	0.3816	1.0000	1.0000	0.0003
0.100%	0.7790	0.3822	1.0000	1.0000	0.0007	0.7772	0.3839	1.0000	1.0000	0.0000	0.7792	0.3820	0.9999	0.9999	0.0008
0.500%	0.7794	0.3817	1.0000	1.0000	0.0000	0.7771	0.3837	1.0000	1.0000	0.0000	0.7795	0.3816	1.0000	1.0000	0.0000
1.000%	0.7787	0.3820	1.0000	1.0000	0.0000	0.7778	0.3832	1.0000	1.0000	0.0000	0.7787	0.3817	1.0000	1.0000	0.0000
poisoning rate (KDD)	RandField					RandValue					Ours				
	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓	AUC↑	Logloss↓	AUA_1↑	AUA_2↑	$Logloss_1$↓
0.001%	0.7643	0.1792	0.9998	0.9998	0.0003	0.7629	0.1813	0.9965	0.9964	0.0338	0.7643	0.1817	0.9989	0.9989	0.0258
0.005%	0.7650	0.1795	0.9998	0.9998	0.0003	0.7628	0.1808	0.9996	0.9995	0.0100	0.7636	0.1824	1.0000	1.0000	0.0012
0.010%	0.7660	0.1790	1.0000	1.0000	0.0000	0.7614	0.1808	0.9998	0.9998	0.0031	0.7631	0.1818	1.0000	1.0000	0.0008
0.100%	0.7646	0.1790	1.0000	1.0000	0.0000	0.7626	0.1806	1.0000	1.0000	0.0006	0.7643	0.1825	1.0000	1.0000	0.0000
0.500%	0.7676	0.1787	1.0000	1.0000	0.0000	0.7620	0.1805	1.0000	1.0000	0.0002	0.7612	0.1819	1.0000	1.0000	0.0000
1.000%	0.7662	0.1790	1.0000	1.0000	0.0000	0.7620	0.1808	1.0000	1.0000	0.0000	0.7626	0.1822	1.0000	1.0000	0.0000

mation of feature representations learned by the deep model. Then, Beatrix can detect poisoned samples by identifying deviated anomalies from statistical patterns derived from clean samples. Specifically, Beatrix estimates an anomaly threshold of deviations through Gramian information of clean samples.

Following the settings in Beatrix, we randomly select 30 clean samples per label to determine an appropriate threshold. We then craft 4000 samples from each dataset for evaluation, half of which are poisoned samples carrying BAD-FM triggers. As shown in Table 6, Beatrix can only achieve $\leq$ 9.55% TPR @ 5% FPR and $\leq$ 4.70% TPR @ 1% FPR on BAD-FM. The resistance to Beatrix further indicates that defenses designed for DNNs are insufficient for BAD-FM, which targets both deep and shallow components of CTR models.

V. Discussion

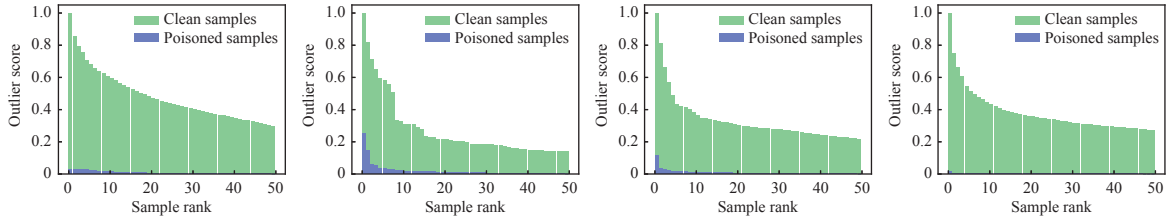
1. Impact of trigger

We evaluate the impact of the trigger size and the trigger composition on the attack performance.

Figure 1 shows an ideal condition for the adversary where even the one-field trigger can yield good performance. Therefore, we gradually reduce the trigger size and compare BAD-FM with a random (i.e., BL1) trigger. We endeavor to use the two representative methods to investigate the general impact of the trigger size. For the datasets with an average of 25 fields, we set the 4-field trigger as an upper bound. Otherwise, the larger trigger will increase the risk of detection. We have set an appropriate poisoning ratio beforehand to shield its possible effects. As shown in Table 7, BAD-FM achieves an ideal

Table 5 Apply model pruning defense on BAD-FM

Dataset	Pruning rate	AUC	Logloss	AUA_1	AUA_2	$Logloss_1$
HUAWEI	10%	0.8059	0.1220	0.9984	0.9984	0.0303
	20%	0.8059	0.1220	0.9984	0.9984	0.0303
	30%	0.8059	0.1220	0.9984	0.9984	0.0303
	40%	0.8059	0.1220	0.9984	0.9984	0.0303
	50%	0.8045	0.1224	0.9984	0.9984	0.0303
	60%	0.7964	0.1250	0.9982	0.9984	0.0313
	70%	0.7870	0.1296	0.9973	0.9972	0.0398
	80%	0.7807	0.1365	0.9957	0.9957	0.0454
	90%	0.7728	0.1751	0.9943	0.9942	0.0462
Dataset	Pruning rate	AUC	Logloss	AUA_1	AUA_2	$Logloss_1$
Criteo	10%	0.7925	0.4611	0.9999	0.9999	0.0003
	20%	0.7925	0.4611	0.9999	0.9999	0.0003
	30%	0.7925	0.4611	0.9999	0.9999	0.0003
	40%	0.7925	0.4611	0.9999	0.9999	0.0003
	50%	0.7914	0.4629	0.9999	0.9999	0.0003
	60%	0.7817	0.4987	0.9999	0.9999	0.0003
	70%	0.7709	0.5191	0.9999	0.9999	0.0004
	80%	0.7625	0.5235	0.9998	0.9998	0.0004
	90%	0.7479	0.5374	0.9996	0.9996	0.0018
Dataset	Pruning rate	AUC	Logloss	AUA_1	AUA_2	$Logloss_1$
Avazu	10%	0.7790	0.3822	0.9974	0.9969	0.0308
	20%	0.7790	0.3822	0.9974	0.9969	0.0308
	30%	0.7790	0.3822	0.9974	0.9969	0.0308
	40%	0.7790	0.3822	0.9974	0.9969	0.0308
	50%	0.7790	0.3822	0.9974	0.9969	0.0308
	60%	0.7662	0.3976	0.9972	0.9969	0.0308
	70%	0.7371	0.4399	0.9965	0.9958	0.0330
	80%	0.7028	0.4874	0.9961	0.9953	0.0398
	90%	0.7022	0.5366	0.9890	0.9870	0.1229
Dataset	Pruning rate	AUC	Logloss	AUA_1	AUA_2	$Logloss_1$
KDD	10%	0.7625	0.1790	0.9999	0.9999	0.0002
	20%	0.7645	0.1790	0.9999	0.9999	0.0002
	30%	0.7645	0.1790	0.9999	0.9999	0.0002
	40%	0.7645	0.1790	0.9999	0.9999	0.0002
	50%	0.7645	0.1790	0.9999	0.9999	0.0002
	60%	0.7508	0.2007	0.9999	0.9999	0.0012
	70%	0.7255	0.2122	0.9999	0.9999	0.0020
	80%	0.7119	0.2255	0.9995	0.9995	0.0082
	90%	0.6680	0.2571	0.9989	0.9988	0.0200



(a) HUAWEI (b) Criteo (c) Avazu (d) KDD

Figure 3 Outlier scores of samples generated by spectral signatures.

Table 6 Defense performance of Beatrix. We test true positive rates (TPR) of detecting poisoned samples under low false positive rates (FPR).

Metrics	Dataset			
	HUAWEI	Criteo	Avazu	KDD
TPR @ 5% FPR	1.95%	8.65%	5.20%	9.55%
TPR @ 1% FPR	1.75%	4.70%	0.80%	2.75%

attack performance even with the smallest trigger.

The features of tabular data may be sparse or dense. We start with a sparse-only trigger and gradually replace a sparse field with a dense one until it becomes fully dense. Since the trigger fields of BAD-FM are dependent on the selection design, we mainly discuss the random (i.e., BL1) trigger, whose field type can be optional. As shown in Table 8, the sparse features dominate, and the dense features have little impact on the performance. The possible reason is that dense features primarily affect the deep component [29] but not the shallow component, and a dense feature only represents a single value, while a sparse feature is condensed to a vector.

2. Trigger robustness

We explore whether the generated trigger is robust to backdoor a DeepFM model with a different deep component (i.e., with different numbers of hidden layers and different network structures). As shown in Table 9, the trigger can still maintain the attack ability when injected to a model with more hidden layers. Given a model of 3 hidden layers and 600 neurons, we develop four different deep component structures: uniform (200-200-200), increasing (100-200-300), decreasing (300-200-100), and diamond (150-300-150). As shown in Table 10, when the structure of the target model is the same as or in the same family as the original model, the trigger can maintain a high attack performance. The results imply that even with limited knowledge and control of the target model, BAD-FM can be effective through data poisoning with a generated trigger.

3. Extension to other CTR models

We extend BAD-FM to two more representative models, i.e., xDeepFM [13] and DCN-Mix [21].

Table 7 The impact of the number of selected fields. The poisoning ratio is fixed as 0.001%

Field size	HUAWEI									
	AUC		Logloss		AUA_1		AUA_2		$Logloss_1$	
	Random	Ours	Random	Ours	Random	Ours	Random	Ours	Random	Ours
1	0.8069	0.8051	0.1222	0.1228	0.9989	0.9998	0.9989	0.9998	0.1372	0.0073
2	0.8068	0.8051	0.1221	0.1228	0.9993	1.0000	0.9993	1.0000	0.0248	0.0011
3	0.8068	0.8049	0.1222	0.1228	0.9998	0.9999	0.9998	0.9999	0.0111	0.0024
4	0.8067	0.8048	0.1222	0.1228	0.9994	1.0000	0.9994	1.0000	0.0232	0.0000
Field size	Criteo									
	AUC		Logloss		AUA_1		AUA_2		$Logloss_1$	
	Random	Ours	Random	Ours	Random	Ours	Random	Ours	Random	Ours
1	0.7971	0.7973	0.4540	0.4539	0.8519	0.9479	0.8173	0.9330	0.8560	0.4212
2	0.7960	0.7972	0.4549	0.4540	0.9390	0.9718	0.9248	0.9639	0.4126	0.2321
3	0.7972	0.7963	0.4540	0.4548	0.9624	0.9907	0.9515	0.9880	0.3507	0.0799
4	0.7971	0.7961	0.4540	0.4550	0.9927	0.9970	0.9904	0.9962	0.1254	0.0236
Field size	Avazu									
	AUC		Logloss		AUA_1		AUA_2		$Logloss_1$	
	Random	Ours	Random	Ours	Random	Ours	Random	Ours	Random	Ours
1	0.7789	0.7789	0.3842	0.3816	0.9969	0.9952	0.9963	0.9942	0.0326	0.0468
2	0.7780	0.7794	0.3828	0.3818	0.9990	0.9990	0.9989	0.9988	0.0056	0.0066
3	0.7779	0.7789	0.3832	0.3820	0.9984	0.9993	0.9981	0.9991	0.0177	0.0051
4	0.7788	0.7791	0.3843	0.3816	0.9995	0.9997	0.9993	0.9996	0.0054	0.0033
Field size	KDD									
	AUC		Logloss		AUA_1		AUA_2		$Logloss_1$	
	Random	Ours	Random	Ours	Random	Ours	Random	Ours	Random	Ours
1	0.7673	0.7677	0.1811	0.1790	0.8392	0.9769	0.8311	0.9757	2.1978	0.3695
2	0.7687	0.7647	0.1803	0.1792	0.9330	0.9998	0.9305	0.9998	1.2357	0.0027
3	0.7686	0.7670	0.1806	0.1788	0.9991	0.9996	0.9990	0.9996	0.0198	0.0059
4	0.7645	0.7665	0.1806	0.1805	0.9960	0.9999	0.9959	0.9999	0.0543	0.0017

Table 8 The impact of field type on the random trigger. The number of the field is fixed as four

Dataset	Combinations	AUC	Logloss	AUA_1	AUA_2	$Logloss_1$
HUAWEI**	Sparse-only	0.8067	0.1222	0.9994	0.9994	0.0232
	Sparse-dominant	0.8069	0.1222	0.9998	0.9998	0.0100
	Equal	0.8069	0.1221	0.9980	0.9980	0.0517
	Dense-dominant	0.8069	0.1222	0.9975	0.9974	0.1746
	Dense-only	0.8067	0.1222	0.5995	0.5909	3.6652
Criteo	Sparse-only	0.7883	0.4664	0.9920	0.9903	0.0720
	Sparse-dominant	0.7883	0.4663	0.9979	0.9973	0.0161
	Equal	0.7879	0.4661	0.9982	0.9978	0.0129
	Dense-dominant	0.7884	0.4660	0.9691	0.9632	0.2207
	Dense-only	0.7874	0.4661	0.5861	0.5853	1.8672

Note: ** Due to their lack of dense features, experiments are not conducted on Avazu and KDD datasets.

Table 9 Trigger robustness in terms of network depth

Dataset	# of layers	AUC	Logloss	AUA_1	AUA_2	$Logloss_1$
HUAWEI	2-layer**	0.8049	0.1228	0.9997	0.9996	0.0303
	3-layer	0.8048	0.1228	0.9999	0.9999	0.0173
	5-layer	0.8044	0.1230	0.9981	0.9981	0.0825
Criteo	2-layer	0.7962	0.4550	0.9984	0.9979	0.0146
	3-layer	0.7958	0.4555	0.9988	0.9985	0.0145
	5-layer	0.7968	0.4544	0.9991	0.9988	0.0159
Avazu	2-layer	0.7791	0.3816	0.9997	0.9996	0.0033
	3-layer	0.7794	0.3820	0.9997	0.9996	0.0036
	5-layer	0.7801	0.3816	0.9964	0.9956	0.0468
KDD	2-layer	0.7643	0.1817	0.9989	0.9989	0.0258
	3-layer	0.7672	0.1788	0.9999	0.9999	0.0030
	5-layer	0.7679	0.1784	0.9996	0.9995	0.0126

Note: ** x-layer means the number of fully-connected layers in the deep component of the target network is x. The source has two layers.

xDeepFM xDeepFM integrates CIN with DNN. CIN aims to generate multiple feature maps, each encoding all the pairwise interactions between features at the current and input levels, therefore sharing some functionalities with CNN and RNN.

DCN-Mix DCN combines the cross network (CrossNet) and DNN. CrossNet learns explicit feature interactions of the input through cross layers, while DCN-Mix leverages a mixture of low-rank cross layers to balance model accuracy and efficiency.

Unlike DeepFM, xDeepFM and DCN-Mix model the high-order feature interactions in an explicit manner and use learnable parameters for automatic training. To extend BAD-FM to the two models, we make some adjustments to alleviate such a gap. For trigger formation, we inherit the abstraction of the meta-node and follow similar gradient optimization steps regardless of the specific operations inside the model. For model retraining, the number of iterations has been adjusted to better memorize the trigger. Since parameterized networks have replaced shallow components, additional resources are required for model retraining. As shown in Table 11, the BAD-FM can be successfully applied to the two models without considerable performance degradation.

Table 10 Trigger robustness in terms of network shape

Dataset	DNN shape	AUC	Logloss	AUA_1	AUA_2	$Logloss_1$
HUAWEI	Constant	0.8048	0.1228	0.9998	0.9998	0.0150
	Increasing	0.8041	0.1232	0.9999	0.9999	0.0205
	Decreasing	0.8046	0.1230	1.0000	1.0000	0.0082
	Diamond	0.8047	0.1229	0.9998	0.9998	0.0366
Criteo	Constant	0.7960	0.4554	0.9972	0.9964	0.0263
	Increasing	0.7946	0.4565	0.9985	0.9981	0.0148
	Decreasing	0.7963	0.4549	0.9981	0.9976	0.0228
	Diamond	0.7952	0.4559	0.9984	0.9979	0.0183
Avazu	Constant	0.7789	0.3831	0.9990	0.9988	0.0820
	Increasing	0.7774	0.3832	0.9946	0.9936	0.1101
	Decreasing	0.7794	0.3821	0.9996	0.9996	0.0034
	Diamond	0.7785	0.3829	0.9978	0.9974	0.0162
KDD	Constant	0.7679	0.1790	0.9951	0.9949	0.0466
	Increasing	0.7639	0.1804	1.0000	1.0000	0.0023
	Decreasing	0.7674	0.1791	1.0000	1.0000	0.0011
	Diamond	0.7651	0.1794	0.9774	0.9762	0.3462

Table 11 Extend BAD-FM to other CTR models. The number of selected fields is fixed as four and the poisoning ratio is fixed as 0.001%

Dataset	xDeepFM				
	AUC	Logloss	AUA_1	AUA_2	$Logloss_1$
HUAWEI	0.8055	0.1221	0.9999	0.9999	0.5511
Criteo	0.7936	0.4600	0.9999	0.9999	0.3576
Avazu	0.7804	0.3812	0.9984	0.9981	0.0414
KDD	0.7643	0.1792	0.9992	0.9991	0.0198
Dataset	DCN-Mix				
	AUC	Logloss	AUA_1	AUA_2	$Logloss_1$
HUAWEI	0.8062	0.1220	0.9999	0.9999	0.0333
Criteo	0.7943	0.4601	0.9999	0.9999	0.0004
Avazu	0.7772	0.3844	0.9969	0.9962	0.0365
KDD	0.7598	0.1800	0.9998	0.9998	0.0112

VI. Related Work

Backdoor attacks Backdoor attacks aim to inject backdoors into DNNs during training such that the backdoored model misclassifies any input with a special trigger while preserving accurate predictions for clean inputs. The trigger in backdoor attacks is a key factor. Triggers can be randomly selected (e.g., a logo or a sticker) [1], [16], [30] or generated based on the model structure [2], [15], [31]. Compared with random triggers, model-dependent triggers are more complicated but attain a

better attack performance. As far as we are concerned, all existing backdoor attacks are designed for image-, voice-, or speech-related tasks. The pioneering work BadNets [1] adopted a random trigger and retrained the model with trigger-attached images for backdoor injection. Salem *et al.* [30] proposed a dynamic random trigger generation method based on a generative network. Model-dependent triggers are first proposed by Liu *et al.* [2], in which the trigger is generated to strongly activate a selected neuron in the model. Gong *et al.* [15] proposed to consider both weight and activation for neuron selection. Zhao *et al.* [3] proposed the first backdoor attack for video classification tasks, generating triggers the same way as the universal adversarial patch. Zhai *et al.* [32] designed a backdoor attack against speaker verification tasks by randomly selecting an utterance as the trigger. For natural language processing tasks, backdoor attacks have been developed using a long neutral sentence [33] or a rare word [34], [35] as the trigger. A few studies [24], [36] evaluated the backdoor effects on tabular data, but only concentrated on the simple DNN with fully connected layers. Unlike these existing works, our attack targets a tabular data scenario and identifies the vulnerability in the CTR prediction model.

Backdoor defenses A series of defense strategies have been proposed to defend against backdoor attacks by detecting whether the input contains a trigger [27], [37], [38] or the model contains a backdoor [39], [40]. Gao *et al.* [38] proposed a strong intentional perturbation based system that copies the incoming input multiple times, then perturbs each copy with various samples, and observes the randomness of those perturbed samples' predicted labels. In general, a low entropy represents that the target sample contains a backdoor trigger. Tang *et al.* [41] proposed the statistical contamination analyzer (SCAn) to decompose the feature representation of each test sample into a distinct identity component and a common variation component, therefore making it easy to capture the within-class anomaly. Recently, Ma *et al.* [28] proposed Beatrix, which relaxes basic assumptions of SCAn and enables more accurate feature modeling via high-order Gram matrices. In an orthogonal way, Liu *et al.* [39] proposed artificial brain stimulation to scan the target DNN for one malicious neuron and inspect the outcome of the reverse engineered trigger. Xu *et al.* [42] proposed the meta neural trojan detection, which produces diverse backdoored models through jumbo learning and trains a meta-classifier to predict whether the target DNN model is backdoored or not. Due to the aforementioned difference in the feature representation and model structures, most defenses can be hard to apply to our attack settings. We first adopt an attack-agnostic defense, i.e., pruning [26], trying to eliminate the backdoor effect on the deep component. Moreover, we consider two input-wise detection strategies for tabular data, including the representative SS [27] and the sate-of-the-art Beatrix [28]. Our results demonstrate the defense resistance of BAD-FM.

CTR models In recent years, predicting CTR has received much attention in academia and industry [43], [44]. In online advertising, search engines must estimate the CTR accurately and therefore display the ads precisely to maximize the revenue. Modeling feature interactions is the key to the success of CTR prediction. Considering the insufficient first-order interaction modeled by LR [19], more models have been proposed based on FM [9]. Going a step further, field-aware factorization machine [45] considers field-aware interactions, while attentional factorization machine [46] automatically models such weights via attention mechanism. However, these approaches are often regarded as shallow models with low-order information. With the aid of DNN, researchers now follow a general paradigm to concatenate the mapped embedding vectors and feed them into the neural network for higher-order feature interactions. Factorization-machine supported neural network [11] applies DNN to process the feature embeddings initialized by a pre-trained FM. Product-based neural network [47] inserts a product layer between the embedding layer and DNN to enrich the multi-field tabular data interaction. Similarly, neural factorization machine [48] introduces a bi-interaction pooling layer in between, which enhances the FM's expressiveness. More recent works can be generalized to a hybrid structure that combines feature interaction learning tasks with deep networks. DeepFM [22] follows the Wide & Deep [12] to model low- and high-order feature interactions simultaneously but replaces the linear component with the FM. Considering that DNN supplies the high-order feature interaction in an implicit way, some specifically designed networks are carried out to learn in an explicit fashion as supplements. DCN [49] introduces the cross network, which takes the outer product of features in a bit-wise manner, and its efficiency is further improved by DCN-V2 [21]. xDeepFM [13] proposes CIN to generate feature interactions at the vector level. Meanwhile, AutoInt [50] uses the attention mechanism for measuring such correlations, which offers good model explainability. In this paper, we primarily consider backdoor attacks, which pose severe threats to deep learning models. We have demonstrated that our BAD-FM can be successfully applied to multiple deep CTR models while maintaining the attack performance.

VII. Ethical/Societal Impact

Our research may be leveraged by attackers to launch such backdoor attacks against online recommendation systems, which will cause considerable potential economic loss. Since our attack has shown resistance to several possible defenses, exploring effective defense strategies against this attack scenario will attract more social attention. Still, we believe novel and efficient defenses against the proposed attack will soon be introduced after our research is released.

VIII. Conclusion

This paper presents the design, implementation, and evaluation of BAD-FM, a backdoor attack against tabular data prediction models. BAD-FM features systematic trigger generation and backdoor injection algorithms to address various technical challenges, including the crucial differences in feature representation and model structure of tabular data. Extensive experiments have verified the effectiveness of BAD-FM. Furthermore, we endeavor to attack other CTR models within our framework, expanding the applicability of BAD-FM. Such attacks reveal potential security threats to the deep-learning-based CTR prediction paradigm and extend the research area of backdoor attacks.

References

[1] T. Y. Gu, K. Liu, B. Dolan-Gavitt, *et al.*, "BadNets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7 pp. 47230–47244, 2019.

[2] Y. Q. Liu, S. Q. Ma, Y. Aafer, *et al.*, "Trojaning attack on neural networks," in *Proceedings of the 25th Annual Network and Distributed System Security Symposium*, San Diego, USA, pp. 1–15, 2018.

[3] S. H. Zhao, X. J. Ma, X. Zheng, *et al.*, "Clean-label backdoor attacks on video recognition models," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, pp. 14431–14440, 2020.

[4] Y. Q. Liu, G. Y. Shen, G. H. Tao, *et al.*, "Piccolo: Exposing complex backdoors in NLP transformer models," in *Proceedings of the 2022 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, pp. 2025–2042, 2022.

[5] W. Zong, Y. W. Chow, W. Susilo, *et al.*, "TrojanModel: A practical Trojan attack against automatic speech recognition systems," in *Proceedings of the 2023 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, pp. 1667–1683, 2023.

[6] J. R. Qin, W. N. Zhang, R. Su, *et al.*, "Retrieval & interaction machine for tabular data prediction," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, Virtual Event, pp. 1379–1389, 2021.

[7] K. Kireev, B. Kulynych, and C. Troncoso, "Adversarial robustness for tabular data through cost and utility awareness," in *Proceedings of the 30th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, pp. 1–18, 2023.

[8] Q. Pang, Y. Y. Yuan, S. Wang, *et al.*, "ADI: Adversarial dominating inputs in vertical federated learning systems," in *Proceedings of the 2023 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, pp. 1875–1892, 2023.

[9] S. Rendle, "Factorization machines," in *Proceedings of the 2010 IEEE International Conference on Data Mining*, Sydney, Australia, pp. 995–1000, 2010.

[10] R. J. Oentaryo, E. P. Lim, J. W. Low, *et al.*, "Predicting response in mobile advertising with hierarchical importance-aware factorization machine," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, New York, NY, USA, pp. 123–132, 2014.

[11] W. N. Zhang, T. M. Du, and J. Wang, "Deep learning over multi-field categorical data," in *Proceedings of the 38th European Conference on Information Retrieval*, Padua, Italy, pp. 45–57, 2016.

[12] H. T. Cheng, L. Koc, J. Harmsen, *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, Boston, MA, USA, pp. 7–10, 2016.

[13] J. X. Lian, X. H. Zhou, F. Z. Zhang, *et al.*, "xDeepFM: Combining explicit and implicit feature interactions for recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, London, UK, pp. 1754–1763, 2018.

[14] S. F. Li, M. H. Xue, B. Z. H. Zhao, *et al.*, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2088–2105, 2021.

[15] X. L. Gong, Y. J. Chen, Q. Wang, *et al.*, "Defense-resistant backdoor attacks against deep neural networks in outsourced cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2617–2631, 2021.

[16] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, New York, NY, USA, pp. 11957–11965, 2020.

[17] X. Y. Chen, C. Liu, B. Li, *et al.*, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint*, arXiv: 1712.05526, 2017.

[18] J. Y. Lin, L. Xu, Y. Q. Liu, *et al.*, "Composite backdoor attack for deep neural network by mixing existing benign features," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, Virtual Event, pp. 113–131, 2020.

[19] K. C. Lee, B. Orten, A. Dasdan, *et al.*, "Estimating conversion rate in display advertising from past erformance data," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Beijing, China, pp. 768–776, 2012.

[20] X. R. He, J. F. Pan, O. Jin, *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, New York, NY, USA, pp. 1–9, 2014.

[21] R. X. Wang, R. Shivanna, D. Cheng, *et al.*, "DCN V2: Improved deep & cross network and practical lessons for web-scale learning to rank systems," in *Proceedings of the Web Conference*, Ljubljana, Slovenia, pp. 1785–1797, 2021.

[22] H. F. Guo, R. M. Tang, Y. M. Ye, *et al.*, "DeepFM: A factorization-machine based neural network for CTR prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, pp. 1725–1731, 2017.

[23] E. Wenger, J. Passananti, A. N. Bhagoji, *et al.*, "Backdoor attacks against deep learning systems in the physical world," in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, TN, USA, pp. 6202–6211, 2021.

[24] C. L. Xie, K. L. Huang, P. Y. Chen, *et al.*, "DBA: Distributed backdoor attacks against federated learning," in *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, pp. 1–19, 2020.

[25] Y. Zeng, M. Z. Pan, H. A. Just, *et al.*, "Narcissus: A practical clean-label backdoor attack with limited information," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, Copenhagen, Denmark, pp. 771–785, 2023.

[26] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Proceedings of the 21st International Symposium on Research in Attacks, Intrusions, and Defenses*, Heraklion, Greece, pp. 273–294, 2018.

[27] B. Tran, J. Li, and A. Mądry, "Spectral signatures in backdoor attacks," in *Proceedings of the 32nd International Con-*

ference on Neural Information Processing Systems, Montréal, Canada, pp. 8011–8021, 2018.

[28] W. L. Ma, D. R. Wang, R. X. Sun, *et al.*, "The "Beatrix" resurrections: Robust backdoor detection via gram matrices," in *Proceedings of the 30th Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, pp. 1–18, 2023.

[29] Q. Q. Song, D. H. Cheng, H. N. Zhou, *et al.*, "Towards automated neural interaction discovery for click-through rate prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Virtual Event, pp. 945–955, 2020.

[30] A. Salem, R. Wen, M. Backes, *et al.*, "Dynamic backdoor attacks against machine learning models," in *Proceedings of the IEEE 7th European Symposium on Security and Privacy*, Genoa, Italy, pp. 703–718, 2022.

[31] S. Wang, S. Nepal, C. Rudolph, *et al.*, "Backdoor attacks against transfer learning with pre-trained deep learning models," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1526–1539, 2022.

[32] T. Q. Zhai, Y. M. Li, Z. Q. Zhang, *et al.*, "Backdoor attack against speaker verification," in *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, Toronto, Canada, pp. 2560–2564, 2021.

[33] J. Z. Dai, C. S. Chen, and Y. F. Li, "A backdoor attack against LSTM-based text classification systems," *IEEE Access*, vol. 7, pp. 138872–138878, 2019.

[34] S. Garg, A. Kumar, V. Goel, *et al.*, "Can adversarial weight perturbations inject neural backdoors," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, Virtual Event, pp. 2029–2032, 2020.

[35] W. K. Yang, L. Li, Z. Y. Zhang, *et al.*, "Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in NLP models," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online, pp. 2048–2058, 2021.

[36] M. Jagielski, G. Severi, N. P. Harger, *et al.*, "Subpopulation data poisoning attacks," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, Virtual Event, pp. 3104–3122, 2021.

[37] E. Chou, F. Tramèr, and G. Pellegrino, "Sentinet: Detecting localized universal attacks against deep learning systems," in *Proceedings of the 2020 IEEE Security and Privacy Workshops*, San Francisco, CA, USA, pp. 48–54, 2020.

[38] Y. S. Gao, C. E. Xu, D. R. Wang, *et al.*, "STRIP: A defence against Trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, San Juan, PR, USA, pp. 113–125, 2019.

[39] Y. Q. Liu, W. C. Lee, G. H. Tao, *et al.*, "ABS: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London, UK, pp. 1265–1282, 2019.

[40] B. L. Wang, Y. S. Yao, S. Shan, *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, pp. 707–723, 2019.

[41] D. Tang, X. F. Wang, H. X. Tang, *et al.*, "Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection," in *Proceedings of the 30th USENIX Security Symposium*, Virtual Event, pp. 1541–1558, 2021.

[42] X. J. Xu, Q. Wang, H. C. Li, *et al.*, "Detecting AI Trojans using meta neural analysis," in *Proceedings of the 2021 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, pp. 103–120, 2021.

[43] B. Sarwar, G. Karypis, J. Konstan, *et al.*, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, Hong Kong, China, pp. 285–295, 2001.

[44] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, Boston, MA, USA, pp. 191–198, 2016.

[45] Y. Juan, Y. Zhuang, W. S. Chin, *et al.*, "Field-aware factorization machines for CTR prediction," in *Proceedings of the 10th ACM Conference on Recommender Systems*, Boston, MA, USA, pp. 43–50, 2016.

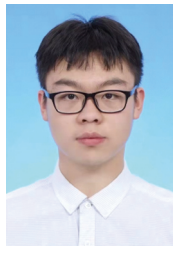
[46] J. Xiao, H. Ye, X. N. He, *et al.*, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, pp. 3119–3125, 2017.

[47] Y. R. Qu, H. Cai, K. Ren, *et al.*, "Product-based neural networks for user response prediction," in *Proceedings of the IEEE 16th International Conference on Data Mining*, Barcelona, Spain, pp. 1149–1154, 2016.

[48] X. N. He and T. S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Shinjuku, Japan, pp. 355–364, 2017.

[49] R. X. Wang, B. Fu, G. Fu, *et al.*, "Deep & cross network for ad click predictions," in *Proceedings of the ADKDD'17*, Halifax, Canada, article no. 12, 2017.

[50] W. P. Song, C. C. Shi, Z. P. Xiao, *et al.*, "AutoInt: Automatic feature interaction learning via self-attentive neural networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, Beijing, China, pp. 1161–1170, 2019.



Lingshuo MENG received the B.E. degree in cyber science and engineering from Wuhan University, Wuhan, China, in 2022. He is currently pursuing the M.S. degree with the College of Electrical Engineering, Zhejiang University, Hangzhou, China. His research interests include network security and AI security. (Email: emeng@zju.edu.cn)

Xueluan GONG received ther B.S. degree in computer science and electronic engineering from Hunan University, Changsha, China, in 2018. She is currently pursuing the Ph.D. degree in computer science with Wuhan University, Wuhan, China. Her research interests include AI security and information security. (Email: xueluangong@whu.edu.cn)

Yanjiao CHEN received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2010 and Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, China, in 2015. She is currently a Bairen researcher with the College of Electrical Engineering, Zhejiang University, Hangzhou, China. Her research interests include computer networks, network security, and Internet of things. (Email: chenyanjiao@zju.edu.cn)