mathematics

MDPI

Article

A Sampling-Based Method for Detecting Data Poisoning Attacks in Recommendation Systems

Mohan Li, Yuxin Lian, Jinpeng Zhu, Jingyi Lin, Jiawen Wan and Yanbin Sun *

Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China; limohan@gzhu.edu.cn (M.L.); lianyuxin@e.gzhu.edu.cn (Y.L.); zhujinpeng@e.gzhu.edu.cn (J.Z.); linjingyi1211@e.gzhu.edu.cn (J.L.); wanjiawen@e.gzhu.edu.cn (J.W.)
* Correspondence: sunyanbin@gzhu.edu.cn

Abstract: The recommendation algorithm based on collaborative filtering is vulnerable to data poisoning attacks, wherein attackers can manipulate system output by injecting a large volume of fake rating data. To address this issue, it is essential to investigate methods for detecting systematically injected poisoning data within the rating matrix. Since attackers often inject a significant quantity of poisoning data in a short period to achieve their desired impact, these data may exhibit spatial proximity. In other words, poisoning data may be concentrated in adjacent rows of the rating matrix. This paper capitalizes on the proximity characteristics of poisoning data in the rating matrix and introduces a sampling-based method for detecting data poisoning attacks. First, we designed a rating matrix sampling method specifically for detecting poisoning data. By sampling differences obtained from the original rating matrix, it is possible to infer the presence of poisoning attacks and effectively discard poisoning data. Second, we developed a method for pinpointing malicious data based on the distance of rating vectors. Through distance calculations, we can accurately identify the positions of malicious data. After that, we validated the method on three real-world datasets. The results demonstrate the effectiveness of our method in identifying malicious data within the rating matrix.

Keywords: data poisoning; recommendation systems; ensemble learning; data poisoning detection

MSC: 68M25

Citation: Li, M.; Lian, Y.; Zhu, J.; Lin, J.; Wan, J.; Sun, Y. A Sampling-Based Method for Detecting Data Poisoning Attacks in Recommendation Systems. *Mathematics* **2024**, *12*, 247. https://doi.org/10.3390/math12020247

Academic Editors: Jun Feng, Changqing Luo and Mamoun Alazab

Received: 21 December 2023
Revised: 8 January 2024
Accepted: 9 January 2024
Published: 12 January 2024

1. Introduction

Recommendation systems play a pivotal role in personalized applications, including e-commerce, film, and music platforms [1–3]. Despite their significance, the current recommendation systems often fall short of adequately addressing security concerns. Recent research highlights the pronounced susceptibility of recommendation systems to data poisoning attacks, wherein attackers deliberately inject a substantial volume of fake ratings to disrupt authentic data distribution. This intentional manipulation distorts the recommendation results in alignment with the attacker's objectives.

The consequences of recommendation algorithms succumbing to poisoning attacks extend beyond mere financial losses for businesses and users. Such vulnerabilities create an environment conducive to harmful activities, including the spread of rumors, the promotion of cults, and the compromise of social stability and public safety. Recognizing and mitigating these security issues is paramount for safeguarding the integrity and reliability of recommendation systems in diverse application domains [4,5].

Traditional machine learning methods often assume that input data are reliable and error-free. However, this assumption becomes a vulnerability that attackers can exploit to intentionally disrupt data distribution, contaminate training data, and manipulate models. By injecting errors into the input data, attackers can skew the learning process, leading to a misalignment between the model and the genuine underlying patterns in the data. Consequently, this manipulation results in reduced model accuracy and compromised

performance [1–3,6]. This attack is known as a data poisoning attack. Research on data poisoning attacks often draws inspiration from real-life scenarios where attackers exploit vulnerabilities in recommendation systems for various profit-driven motives. On the Internet, attackers may target recommendation systems, leveraging the user feedback mechanism to inject malicious data intentionally. This injection of manipulated data disrupts the model training process, leading the algorithm to make recommendations that deviate from what would be genuinely beneficial for normal users. The goal of such attacks is to skew the recommendation results to serve the attacker's objectives [6].

Many studies have shown that recommendation algorithms based on collaborative filtering are susceptible to data poisoning attacks [7]. Due to the inherent challenge of the system in distinguishing between data from fake users and real users, attackers can exploit this vulnerability by injecting a specific volume of fake users. By modifying the content and distribution of the training data, they can influence the training outcomes of the model. Consequently, the recommendation system may generate suggestions aligning with the attacker's intentions [8]. For example, attackers can maximize the effectiveness and credibility of the system by carefully forging users, increasing or decreasing the relevance of target items to specific users or items, thereby purposefully improving or suppressing the recommendation of target items. This kind of attack greatly affects the credibility of the recommendation system [6].

Attackers often find it relatively easy to execute data poisoning attacks by either creating fraudulent accounts or acquiring trolls. However, defenders face significant challenges as they lack prior knowledge about when an attack will occur, the specific target, or the motives behind the poisoning. The sheer volume of data compounds the difficulty of implementing targeted defenses effectively [9].

Defenses against data poisoning attacks in recommendation systems predominantly focus on user profiles [10] with explicit semantics or fabricated product reviews [11]. This involves analyzing user profiles for homology errors or evaluating product reviews to identify machine-generated content. Such approaches aim to uncover potential poisoning data. However, it may not be easy to obtain these user profiles or product reviews. Therefore, whether it is possible to detect poisoning attacks by directly detecting anomalies in product rating data is an issue worth studying.

In this paper, we propose a series of detection methods for identifying potential malicious rating data in recommender systems. Our approach involves sampling the suspected contaminated dataset and attempting to replace its distribution with the sample distribution derived from n samplings. If the distributions of the n sampled datasets are consistent, it suggests the dataset is likely clean. However, if there is a discernible abnormal distribution, it raises the suspicion of contamination from a poisoning attack. The contributions of this paper are as follows.

1. We developed a sampling-based method for detecting malicious rating data. By conducting multiple samplings, we assessed the influence of the sampled data on the recommendation algorithm and scrutinized the presence of malicious ratings within the dataset.
2. We devised a method for pinpointing the location of malicious data using scoring vector distance. Through distance calculations, we can accurately identify the presence and position of malicious data within the sampled dataset.
3. We validated the effectiveness of the proposed algorithm across various attack models and algorithm parameter configurations. Experimental results demonstrate that our method effectively detects malicious scoring data.

The subsequent sections of this paper are structured as follows. Section 2 provides an overview of related research. Section 3 presents the sampling-based method for detecting malicious data. Section 4 introduces the method for locating malicious data based on scoring vector distance. Section 5 outlines the experimental results, and Section 6 concludes the paper.

2. Related Work

Data poisoning has attracted attention in various fields in recent years. Notably, ref. [12] presents an optimization method for data poison attacks on self-regression models. In [13], a test method for the robustness of the knowledge graph is introduced. Ref. [8] designed a targeted data poison attack. The data poison attack of the recommendation system is relatively easy to achieve and can obtain direct economic benefits, which has attracted widespread attention from domestic and foreign researchers [14].

In the realm of recommendation system data poisoning attacks, the earliest instances of shilling attacks or data injection attacks are documented in [15]. These attacks aim to manipulate the recommendation of target items based on the attacker's interests, either increasing or decreasing their prominence. These attacks aim to manipulate the recommendation system based on the attacker's interests, either promoting or demoting certain items. Specifically, attackers typically register fake users in the system and then control the rating profiles of each fake user. They assign carefully designed scores to selectively chosen items, injecting the system with manipulated data to achieve the goals of the poisoning attack. Subsequently, various generic data poisoning attack techniques may be employed. The two attack methods proposed in [15,16] are random attack and average attack, where each fake user randomly selects items for rating. Ref. [17] introduces the popularity attack, targeting relatively popular items for rating. Unlike random attacks, popularity attacks choose a subset of popular items and randomly select others as fillers. The aforementioned data poisoning attacks all involve generating fake users using manual rules without being restricted to specific recommendation algorithms [18].

In recent years, the focus on optimizing data poisoning has shifted towards attacks targeting specific types of recommendation systems. Nearest neighbor recommendation algorithms primarily utilize the similarity between users or items for recommendation [19,20]. Ref. [21] introduces a data poisoning attack for these nearest neighbor-based recommendation algorithms. Building on shilling attacks and assuming incomplete datasets, ref. [22] proposes a novel attack method to address the incompleteness and perturbation of user rating data. Ref. [23] presents a model to balance the unidentifiability and malice of rating data.

The focus of poisoning attacks on matrix factorization [24] lies in the robustness of matrix completion, as a small number of factors in the latent factor matrix may lead to arbitrary perturbations in the results [25,26]. The data poisoning attack model proposed in [27], based on matrix factorization recommendation algorithms, introduces a synthetic user-item rating matrix into the model through matrix factorization. They intercept and synthesize the completion matrix, increasing the Root Mean Square Error (RMSE) after poisoning. Subsequently, they train a new user-item rating matrix and inject it into the matrix factorization collaborative filtering recommendation model to attack it, disrupting the model's output. The data poisoning attack on cross-domain recommendations, as presented in [28], is an extension of the approach outlined in [27]. Leveraging the associations and characteristic information existing between different domains, they conduct matrix factorization operations separately in the target and auxiliary domains. By using shared items between the target and auxiliary domains, the item latent factor matrices act as connectors, linking the two domains. By controlling the RMSE between the two domains, they train the poisoning matrix, disrupting the recommendation results to achieve the poisoning objectives.

In addition, the pseudo-co-access injection attack proposed in [29] is aimed at injection attacks on recommendation systems based on association rules. This attack describes pseudo-co-access injection as a constrained linear optimization problem. By solving this problem, attackers can carry out maximally threatening attacks. Ref. [30] introduces data poisoning attacks targeting graph recommendation systems. They transform the poisoning attack problem into an optimization problem, specifically the problem of maximizing the hit rate of the target item, to enhance the attack's effectiveness.

Simultaneously, with the development of deep neural networks and the realization that nonlinear transformations in the recommendation process can be achieved through deep learning techniques [31,32], deep learning has gradually become a technological trend in the field of recommendation systems [1,32–34]. However, deep learning algorithms also exhibit vulnerabilities [35]. Ref. [36] presents a data poisoning technique targeting deep recommendation algorithms, transforming the optimization problem into a hit rate maximization problem.

Traditional Intrusion Detection Systems (IDS) [37] deployed on networks or endpoints cannot be directly employed to detect data poisoning attacks. This is because, in the case of data poisoning attacks, attackers do not need to gain access to a particular enterprise's resources through privilege escalation or lateral movement. Instead, they simply use a registered account to log into the system, similar to regular users of the service, and then provide falsified rating data. When the attacker submits a significant amount of fake data, they naturally influence the system's output results. It is challenging to construct an attack graph specifically tailored to data poisoning attacks using the output of certain devices like vulnerability scanners. Hence, methods like attack graphs are not suitable for detecting data poisoning attacks.

In summary, data poisoning attacks on recommendation systems can be categorized based on the method attackers use to generate fake users and the target model. However, effective defense methods are still scarce because defenders typically find it challenging to determine when and with what data attackers launch an attack.

3. Threat Model

The threat model considered in this paper is illustrated in Figure 1. A rating matrix R can be represented as $R_{|U| \times |V|}$, where U is the set comprising all users in the recommendation system, $|U|$ is the total number of users, V is the set comprising all items (i.e., products) in the recommendation system, and $|V|$ is the total number of items. The value of the element in the i-th row and j-th column of R represents the rating given by user u_i to item v_j.

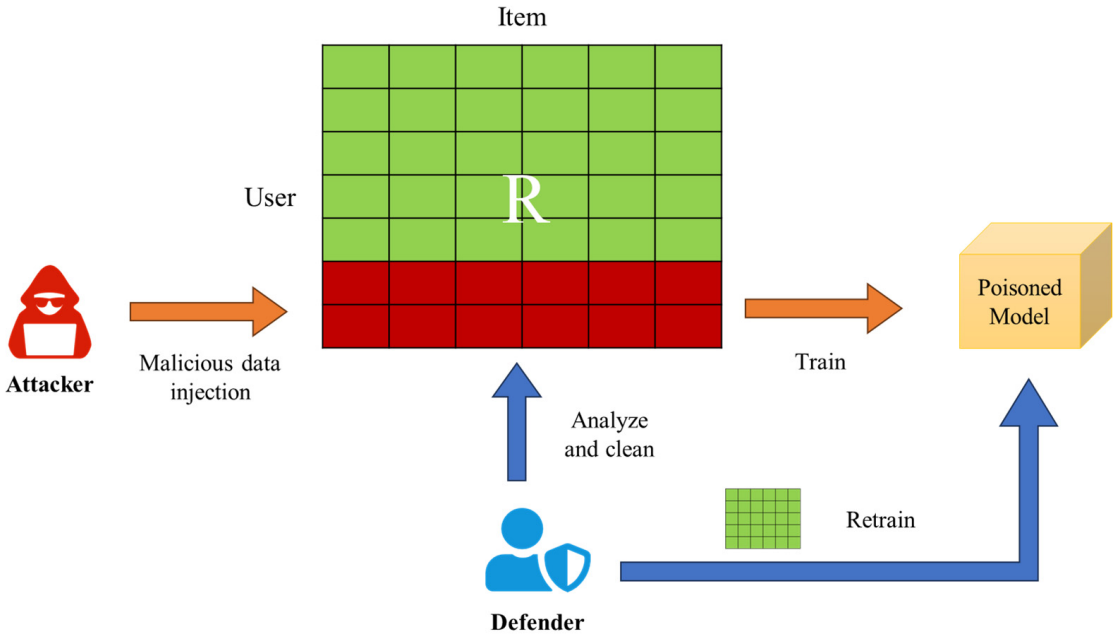


Figure 1. Threat model.

The capabilities of the attacker and defender are as follows.

Attacker: The attacker has the ability to construct fake users and inject fake ratings into the rating matrix, thereby contaminating R. If a recommendation model is trained

using the contaminated data, it becomes a poisoned model. The model will adjust ratings for specific items as per the attacker's settings, either increasing or decreasing them. To ensure the effectiveness of their attacks, attackers often inject attack data into the rating matrix in a short time and in large quantities. As a result, attack data tend to concentrate on adjacent rows in R.

Defender: The defender is unaware of the presence of poisoned data and has access to the complete rating matrix R, which includes both legitimate and injected fake data. By analyzing R, the defender aims to identify the existence of injected fake data. The defender then performs data cleansing, removing malicious entries from R to obtain a clean rating matrix. Subsequently, the defender retrains the model based on the clean ratings to obtain an unbiased recommendation model.

This paper takes the perspective of the defender and attempts to provide effective methods for detecting and locating malicious data. The goal is to assist defenders in identifying the presence of malicious data in R, pinpointing its location, and subsequently removing such data.

4. Detecting Malicious Rating Data Based on Sampling

The overview of the method proposed by us is depicted in Figure 2. We perform n iterations of sampling on the ranking data R, obtaining a sample dataset for each iteration denoted as S_1, S_2, ..., S_n. We conduct a distribution analysis on the results of these n samplings and calculate a suspiciousness score, Suspicious(S_i), for each S_i. Based on Suspicious(S_i), we then detect whether S_i contains abnormal data. When the size of S_i is appropriately chosen, this method can detect all poisoned data in the ranking data R.

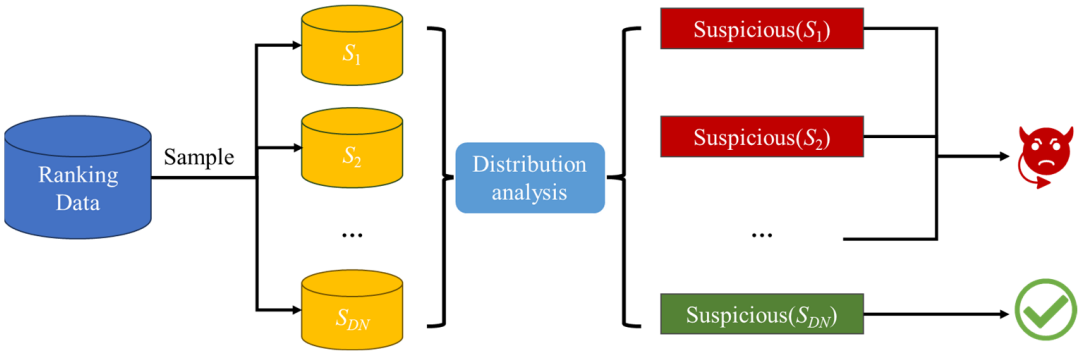


Figure 2. Overview of our method.

For sampling the rating matrix, two methods can be considered: sampling by rows (i.e., users) and sampling by columns (i.e., items). Considering that attackers typically construct fake users to inject fake ratings into R, it is more reasonable to sample by rows.

However, assuming the attacker's poisoning rate is p%, meaning the p% of users in R are injected with fake ratings by the attacker, and if the defender adopts uniform sampling with a sampling rate of q%, the expected number of fake users in each sample is $|R| \times p\% \times q\%$. However, the defender does not know the proportion of attack data and the location of the attack data. Thus, using uniform sampling n times will result in each sample containing approximately $|R| \times p\% \times q\%$ fake users. This method does not manifest significant distribution differences in the sampled data because the p% of malicious users appear in each sampling result.

In reality, for recommendation systems, attackers often register a batch of fake users and inject fake ratings within a short period. Therefore, the distribution of malicious data in R may exhibit clustering characteristics. Based on this idea, we design the sampling method as follows: we sequentially divide R into c partitions and randomly discard d partitions. In some cases, malicious data may be "discarded" in bulk, resulting in a sample with mostly clean data. Figure 3 illustrates the results for $|U| = 1000$, a poisoning rate of 10%, $d = 1$, and

c = {2, 4, 6, 8, 10}. We tested all possible scenarios under the above parameter configuration and calculated the sum of rating predictions for 10 target users when completing the rating matrix R using the latent factor model. The results are shown in Figure 3, with bold results indicating predictions significantly deviating from the average. Since we randomly discard a block of data, and malicious data are a subset that causes significant deviations, in the case of a poisoning rate $p\% << 1$, the event "discarding poisoned data in randomly discarded data" is a low-probability event. Therefore, the part of the data that deviates further from the mean may actually contain fewer poisoned data, while the remaining partitions may lean toward the direction the attacker desires due to the inclusion of more poisoned data. We conducted a detailed examination of the data, and the results confirmed our hypothesis.

c=2	c=4	c=6	c=8	c=10
5000.2391	4475.5002	5489.8818	4759.7034	4935.9686
3819.202	4447.7421	5486.3641	4761.1431	4931.0682
	4494.5036	5490.4479	4753.9494	4927.7595
	3655.7	5495.577	4748.887	4929.1294
		5494.1822	4767.2944	4919.1419
		5304.706	4769.4239	4945.6307
			4749.5632	4939.8203
			4182.653	4949.294
				4939.4887
				4397.474

Figure 3. Sum of predicted ratings on target users. The colored and bold numbers represent the results obtained from the dataset with the least amount of malicious data.

5. Locating Malicious Data Based on Rating Vector Distances

5.1. Suspicious Score Based on Rating Vector Distances

Based on the observations from the previous section, we can see that if malicious data exhibit a certain degree of clustering in the dataset, we can divide the data into blocks, randomly delete several blocks, check the change in the output results of the recommendation algorithm after deletion, and thus, infer the possible locations of malicious data. To characterize the differences in various sampling results more finely, we define a set of target users and observe the differences in the predicted rating vectors for target users under different data subsets to assess the variations between sampling results. We define Suspicious(S_i) as the level of suspicion for the sampling result S_i, with the specific definition as follows:

$$\text{Suspicious}(S_i) = -\Sigma_{j \neq i}\, distance(v_i, v_j)/(c-1) \quad (1)$$

where v_i is the rating vector for the target users obtained by executing the recommendation algorithm on S_i, and $distance(v_i, v_j)$ is the distance between two rating vectors. In the experiments, we tried both Euclidean distance and cosine distance, and the differences in effectiveness were small.

Based on the above idea, we propose a method to discover potential malicious data blocks in the rating matrix. The details are illustrated in Algorithm 1.

Algorithm 1. Calculation of suspicious score.

Input:	Rating matrix R, block parameter c, discard rate d, sampling times DN
Output:	Suspicion scores for each sampling result
1.	Divide R into c blocks along rows
2.	For i = 1 to DN
3.	Randomly discard d blocks from the c blocks to obtain S_i
4.	Execute the recommendation algorithm on S_i to obtain the rating vector v_i for the target users
5.	For i = 1 to DN
6.	Suspicious(S_i) = $-\Sigma_{j \neq i}\, distance(v_i, v_j)/(c-1)$
7.	Return {Suspicious(S_1), . . ., Suspicious(S_{DN})}

The suspicion score Suspicious(S_i) for S_i containing fewer poisoned data is higher than the recommendation results with more poisoned data. Therefore, we only need to return the sample with the lowest Suspicious(S_i) as trustworthy data.

If the detection needs to be completed within a relatively short period, we can directly retain the data blocks with the lowest suspicious scores. Data not belonging to this data block will be considered potential malicious data and discarded. For a more precise identification of malicious data, the parameter c can be adjusted, and Algorithm 1 can be executed multiple times. If it is observed that the suspicious scores of data blocks show an increasing trend followed by a decrease as c increases, the point where the suspicious scores reach their highest may have the least inclusion of normal data in the discarded set. In other words, at this point, the accuracy of locating malicious data is likely the highest among the various tests conducted.

5.2. Determining the Parameters

Given the rating matrix R, as the poisoning rate is unknown, it is challenging to directly determine the appropriate block parameter c, discard rate d, and sampling times DN. In cases where computational control is necessary and the cleanliness of the data is not highly critical, we can make educated guesses about the parameters and directly choose Suspicious(S_i) as the clean data to return. From the experimental results, it can be observed that even with inaccurate guesses, our method can still ensure the return of the cleanest sample among all the samplings.

If high data cleanliness is required, and a certain level of computational cost can be tolerated, you can try different values of c with $d = 1$ and $DN = c$. It is easy to see that, with c given, setting $d = 1$ and $DN = c$ is equivalent to enumerating all possible discard modes. As c increases, the lowest Suspicious(S_i) will exhibit a two-stage pattern: there exists θ such that the lowest Suspicious(S_i) for $c \leq \theta$ is significantly lower than the lowest Suspicious(S_i) for $c > \theta$. By observing the graph of the change in the lowest Suspicious(S_i), if you can find θ that satisfies the above condition, then θ can be considered a suitable value for c and return S_i with the lowest Suspicious(S_i) under this value.

5.3. Discussions

The selection of target users should be consistent and is not recommended to be changed based on sampling results. This is because we need to observe the same set of target users to assess whether the output of the recommendation system is influenced by the ratings of a specific group of users. Additionally, when calculating the similarity of rating vectors, it is essential to compare the differences in recommendation results for the same set of target users under different training data conditions. Under normal circumstances, the recommendation results for target users should not show significant variations based on the presence or absence of a small group of users. If such changes occur, the suspicion level of this small group of users as potential fake users will significantly increase.

The calculation method for the sum of predicted ratings involves directly summing the predicted ratings for target users and comparing the absolute values of the results. This approach is simple and intuitively illustrates the impact of the presence or absence of poisoning data on the recommendation results. However, using this sum directly as a method for calculating abnormal ratings may not be sufficiently robust, as observed in Figure 3. Calculating the cosine or Euclidean distance of rating vectors provides a clearer reflection of these differences. We conducted experiments to explore this aspect as well. Nevertheless, the "sum of predicted ratings" method can easily present the differences among various sampling results in a unified manner, making it convenient for visual analysis and providing security analysts with a visualization tool to assist them in confirming the existence of poisoning data.

Another approach is to use methods such as Jensen–Shannon divergence to directly measure the distribution differences among various sampling results. However, the practical implementation of such methods presents some challenges.

First, the accurate measurement of distribution differences requires a sufficiently large amount of data, but rating matrices are typically very sparse. Moreover, the quantity of poisoning data is often much smaller than the entire rating matrix, making distribution differences less pronounced among different sampling results.

Second, attackers usually have a specific purpose, such as boosting or suppressing the ratings of particular items. To achieve this, they may mimic the average ratings of the general population for other items, making such attacks covert and potentially evading conventional anomaly detection methods. Attackers often exploit subtle distribution differences to cause anomalies in the recommendation results for specific items, which can be more easily observed and analyzed in the recommendation results. Directly measuring distribution differences on the original data may not reveal such anomalies easily.

6. Experimental Results

The experiments were conducted using the MovieLens-1M, Goodbooks, and Anime datasets, with the random attack method selected for poisoning. We evaluated our method in the following aspects. First, we confirmed that our method can effectively detect poisoned data when there is a clustering effect in the poisoned data, and this approach is independent of certain inherent parameters of the algorithm itself. Second, we validated the effectiveness of the suspicious score defined based on rating vector distance in discovering malicious rating data.

6.1. Effectiveness of Sampling

First, we tested the impact of different values of the data block parameter c on a user set with $|U| = 1000$, a poisoning rate $p\%$ of 10%, and $d = 1$ on the MovieLens-1M dataset. The results are shown in Figure 4. In our experiment, we varied c from 2 to 20; in other words, $c = \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$. The results are shown in Figure 4. The vertical axis represents the discarded data block number, and the horizontal axis represents the change in c. Each grid represents the sum of predicted ratings for the target users, with darker colors indicating larger sums. It can be observed that when $c \leq p$, only one block's rating is significantly different from the others. When $c > p$, more than one block's rating starts to differ from the others, but the differences become less pronounced. Therefore, we can also judge the approximate poisoning rate based on the change in ratings as c varies. The sudden increase in abnormal blocks may indicate that c is starting to exceed p.

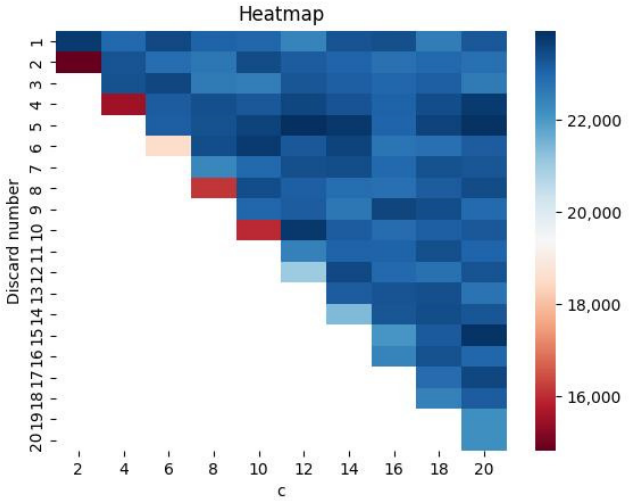


Figure 4. Results under different values of c.

At the same time, we also compared the case of using uniform sampling directly. We sampled 10 times. It is observed that if the sampling rate is low, the sampled data are difficult to accurately use for recommendations, and if the sampling rate is high, the

average proportion of malicious data in each dataset is 10.7%, which is almost equal to the original poisoning rate of 10%. In this case, executing the recommendation algorithm on the sampling results cannot avoid the impact of poisoned data, and the results are similar for each sampled dataset.

The sum of ratings obtained above is represented using a scatter plot in Figure 5. It can be observed that for almost every value of c, there is at least one case where the result significantly deviates from the other data. Since, in our setup, the poisoning rate p% is 10%, it can be seen that when $c \leq p$, the deviations are more pronounced than when $c > p$. Moreover, from our experimental results, it is evident that without knowing whether poisoning exists, setting c to a smaller value allows us to use our method to detect significantly deviating outlier datasets, which may suggest the presence of anomalous data.

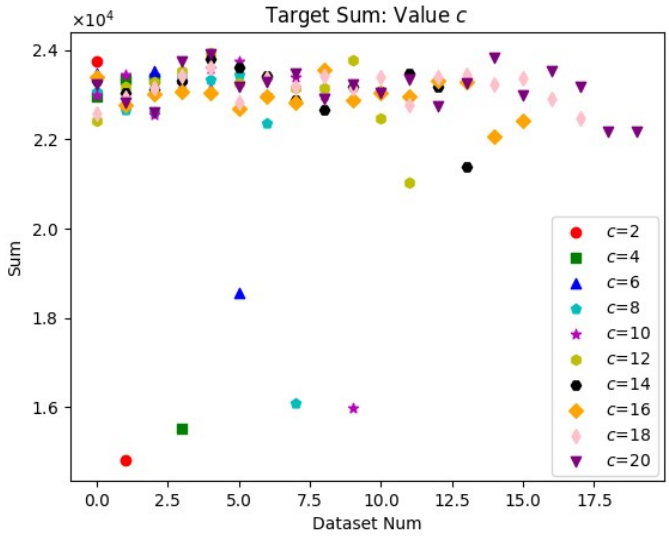
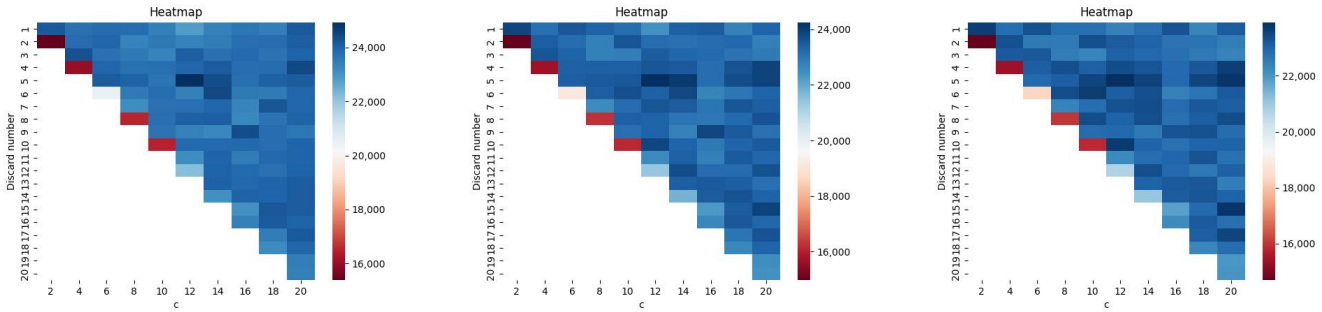


Figure 5. Distribution of summed ratings of target users.

At the same time, we also compared the impact of changing the iteration number e in the LFM algorithm. Generally, a higher number of iterations tends to make the results more robust. However, from the perspective of discovering malicious data, it can be observed that the iteration count has little impact on the effectiveness of our method, as shown in Figure 6. When iterating 2000, 4000, and 6000 times, the experimental results show only slight differences. Even without using any defense measures, a simple random attack is already effective against the recommendation algorithm. Our method accurately captures the differences in the prediction results when $c \leq p$ and $c > p$. In the case of $c \leq p$, the sum of prediction results for the cleanest dataset is shown in red, while for $c > p$, the sum of prediction results is shown in blue.



(**a**) (**b**) (**c**)

Figure 6. Impact of iteration times. (**a**) e = 2000 (**b**) e = 4000 (**c**) e = 6000.

6.2. Suspicious Score

We conducted experiments on the MovieLens-1M, Goodbook, and Anime datasets to evaluate whether our suspicious score (i.e., rating vector distance) is sufficient for distinguishing between clean and poisoned datasets. As observed in the experiments in Section 6.1, using only the sum of predicted ratings can roughly indicate the presence of anomalous data. Our experimental results demonstrate that vector distance provides a clearer reflection of these differences.

We employed cosine distance to measure the dissimilarity between recommendation results. We calculated the distances between the recommendation results on each S_i and those on other S_j, as illustrated in Figures 7–9. From the experimental results, it is evident that when $c \leq p$, the closer c is to p, the further the recommendation results from S_i with the lowest Suspicious(S_i) are from the results of other samples. This is because other datasets contain more malicious data, and their recommendation results are significantly influenced by manipulated malicious data. Recommendation results on clean data are distinctly different from situations involving malicious data. This indicates that using Suspicious(S_i) can effectively distinguish the differences among various sampling results. We also tested the effect of calculating suspicious scores using Euclidean distance. It can be observed that the performance is slightly inferior to cosine distance, but still exhibits a noticeable distinction (Figures 10–12). In the Figures, the x-axis represents the sample dataset identifiers, while the y-axis corresponds to the distances. The blue bars indicate the average distance of the clean datasets, and the yellow bars represent the average distance of the dataset polluted by malicious data.

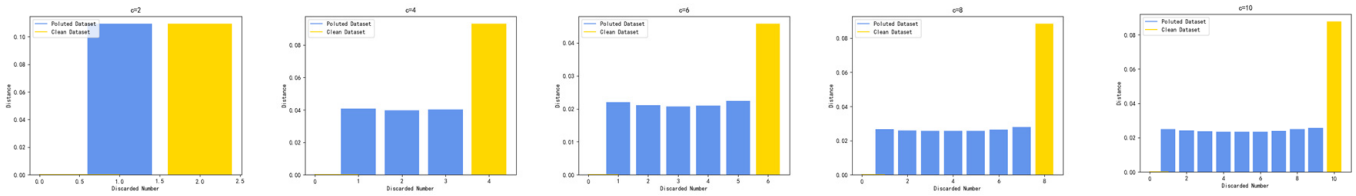


Figure 7. Differences in recommendation results on MovieLens-1M (cosine distance).

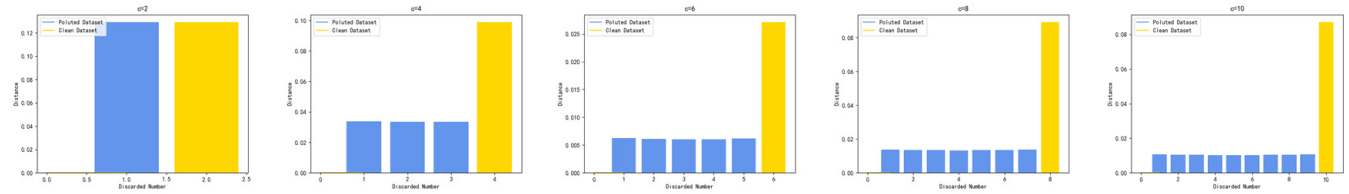


Figure 8. Differences in recommendation results on Goodbook (cosine distance).

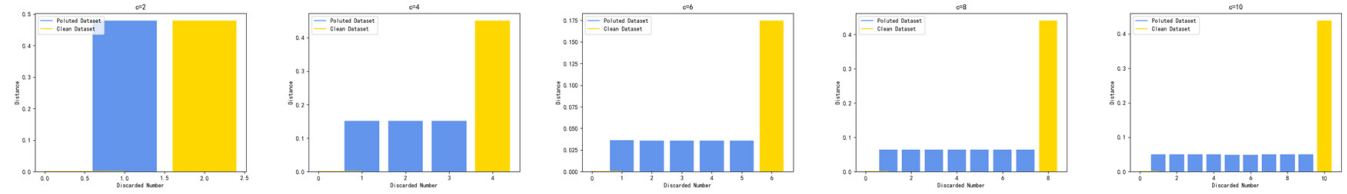


Figure 9. Differences in recommendation results on Anime (cosine distance).

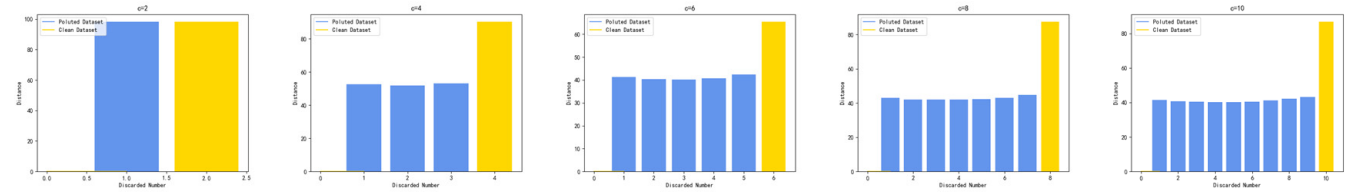


Figure 10. Differences in recommendation results on MovieLens-1M (Euclidean distance).

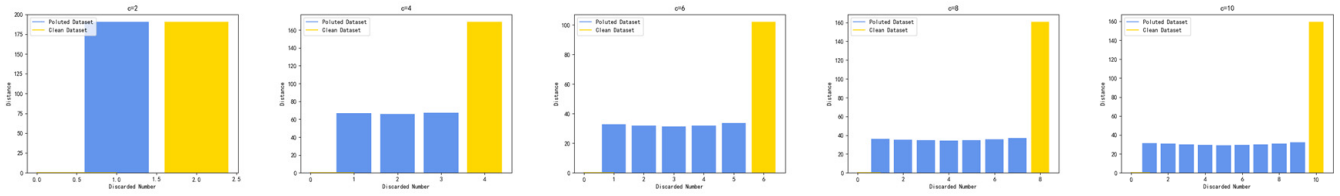


Figure 11. Differences in recommendation results on Goodbook (Euclidean distance).

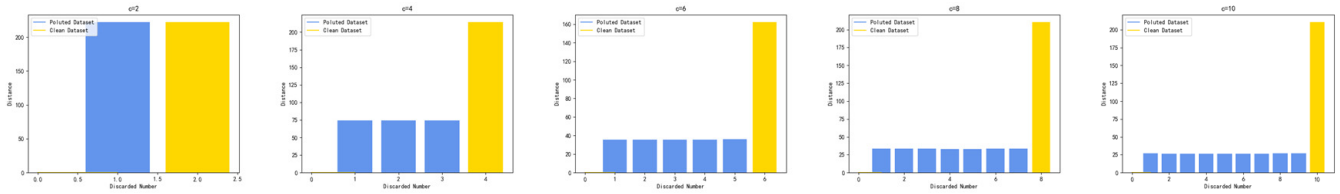


Figure 12. Differences in recommendation results on Anime (Euclidean distance).

Our experimental results on three datasets consistently demonstrate that cosine distance is superior to Euclidean distance. We believe this is because the rating vectors of target users may be similar across many items, with differences only appearing on one or a few specific items intentionally promoted or demoted by the attacker. Cosine distance is more suitable for measuring relative differences in this context. However, since ratings typically have a small range, such as 1 to 5, the absolute value differences in rating vectors are not very large. Therefore, the performance of Euclidean distance as a metric is slightly inferior.

7. Conclusions

This paper studies the issue of detecting poisoning attacks in recommendation systems and proposes a sampling method specifically designed for cases where poisoned data exhibit clustering characteristics. The method aims to identify maliciously poisoned data in the rating matrix. Additionally, a suspiciousness scoring approach based on monitoring the distance of target user rating vectors is introduced. By conducting multiple rounds of sampling and calculating suspiciousness scores, users can infer the presence of anomalous rating data, indicating a potential data poisoning attack. In the event of an attack, the sample with the lowest suspiciousness score can be considered the cleanest, facilitating model retraining to mitigate the impact of malicious data. Experiments conducted on multiple datasets validate the effectiveness of our approach.

Author Contributions: Conceptualization, M.L. and Y.L.; methodology, M.L. and Y.L.; validation, J.Z., J.L. and J.W.; writing—original draft preparation, M.L. and Y.L.; writing—review and editing, M.L. and Y.S.; visualization, Y.L., J.Z., J.L. and J.W.; funding acquisition, M.L. and Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by the National Key Research and Development Plan (Grant No. 2021YFB3101704), the National Natural Science Foundation of China (No. 62372126, 62272119, 62072130, U20B2046), Consulting project of the Chinese Academy of Engineering (No. 2023-JB-13), Guangdong Basic and Applied Basic Research Foundation (No. 2021A1515012307, 2023A1515030142, 2020A1515010450), "National Undergraduate Innovation and Entrepreneurship Training Program" at Guangzhou University (No. 202211078150), Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2019), Guangdong Higher Education Innovation Group (No. 2020KCXTD007), Guangzhou Higher Education Innovation Group (No. 202032854), and Cultivation Project of PZL (No. PZL2022KF0013), Project of Guangzhou University (No. YJ2023047), Guangzhou Basic and Applied Basic Research Foundation (No. SL2023A04J01406).

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Covington, P.; Adams, J.; Sargin, E. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16), Boston, MA, USA, 15–19 September 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 191–198.
2. Schafer, J.B.; Konstan, J.; Riedl, J. Recommender systems in e-commerce. In *World Automation Congress*; IEEE: Piscataway, NJ, USA, 1999.
3. Xie, F.; Li, S.; Chen, L.; Xu, Y.; Zheng, Z. Generative Adversarial Network Based Service Recommendation in Heterogeneous Information Networks. In Proceedings of the 2019 IEEE International Conference on Web Services (ICWS), Milan, Italy, 8–13 July 2019; pp. 265–272.
4. Tian, Z.; Li, M.; Qiu, M.; Sun, Y.; Su, S. Block-DEF: A Secure Digital Evidence Framework using Blockchain. *Inf. Sci.* **2019**, *491*, 151–165. [CrossRef]
5. Li, M.; Sun, Y.; Lu, H.; Maharjan, S.; Tian, Z. Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems. *IEEE Internet Things J.* **2019**, *7*, 6266–6278. [CrossRef]
6. Jia, J.; Liu, Y.; Cao, X.; Gong, N.Z. Certified robustness of nearest neighbors against data poisoning and backdoor attacks. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 9575–9583. [CrossRef]
7. Fang, M.; Gong, N.Z.; Liu, J. Influence function based data poisoning attacks to top-n recommender systems. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 3019–3025.
8. Ma, Y.; Zhu, X.; Hsu, J. Data poisoning against differentially-private learners: Attacks and defenses. *arXiv* **2019**, arXiv:1903.09860.
9. Jia, J.; Cao, X.; Gong, N.Z. Intrinsic certified robustness of bagging against data poisoning attacks. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 7961–7969. [CrossRef]
10. Li, M.; Sun, Y.; Su, S.; Tian, Z.; Wang, Y.; Wang, X. DPIF: A framework for distinguishing unintentional quality problems from potential shilling attacks. *Comput. Mater. Contin.* **2019**, *59*, 331–344. [CrossRef]
11. Yao, Y.; Viswanath, B.; Cryan, J.; Zheng, H.; Zhao, B.Y. Automated crowdturfing attacks and defenses in online review systems. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October 2017; pp. 1143–1158.
12. Verde, L.; Marulli, F.; Marrone, S. Exploring the impact of data poisoning attacks on machine learning model reliability. *Procedia Comput. Sci.* **2021**, *192*, 2624–2632. [CrossRef]
13. Zhang, H.; Zheng, T.; Gao, J.; Miao, C.; Su, L.; Li, Y.; Ren, K. Data poisoning attack against knowledge graph embedding. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 4853–4859.
14. O'Mahony, M.; Hurley, N.; Kushmerick, N.; Silvestre, G. Collaborative recommendation: A robustness analysis. *ACM Trans. Internet Technol.* **2004**, *4*, 344–377. [CrossRef]
15. Gunes, I.; Kaleli, C.; Bilge, A.; Polat, H. Shilling attacks against recommender systems: A comprehensive survey. *Artif. Intell. Rev.* **2014**, *42*, 767–799. [CrossRef]
16. Kapoor, S.; Kapoor, V.; Kumar, R. A review of attacks and its detection attributes on collaborative recommender systems. *Int. J. Adv. Res. Comput. Sci.* **2017**, *8*, 1188. [CrossRef]
17. Burke, R.; Mobasher, B.; Bhaumik, R. Limited knowledge shilling attacks in collaborative filtering systems. In Proceedings of the 3rd International Workshop on Intelligent Techniques for Web Personalization (ITWP 2005), 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), Edinburgh, UK, 1 August 2005; pp. 17–24.
18. Mobasher, B.; Burke, R.; Bhaumik, R.; Williams, C. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans. Internet Technol. (TOIT)* **2007**, *7*, 23. [CrossRef]
19. Bell, R.M.; Koren, Y. Improved neighborhood-based collaborative filtering. In Proceedings of the KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, CA, USA, 12 August 2007; pp. 7–14.
20. Desrosiers, C.; Karypis, G. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 107–144.
21. Chen, L.; Xu, Y.; Xie, F.; Huang, M.; Zheng, Z. Data poisoning attacks on neighborhood-based recommender systems. *Trans. Emerg. Telecommun. Technol.* **2020**, *32*, e3872. [CrossRef]
22. Zhang, H.; Tian, C.; Li, Y.; Su, L.; Yang, N.; Zhao, W.X.; Gao, J. Data Poisoning Attack against Recommender System Using Incomplete and Perturbed Data. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 2154–2164.
23. Wu, C.; Lian, D.; Ge, Y.; Zhu, Z.; Chen, E. Triple Adversarial Learning for Influence based Poisoning Attack in Recommender Systems. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 1830–1840.
24. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
25. Chen, Y.; Xu, H.; Caramanis, C.; Sanghavi, S. Robust matrix completion and corrupted columns. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), Bellevue, WA, USA, 28 June–2 July 2011; pp. 873–880.
26. Chen, Y.; Jalali, A.; Sanghavi, S.; Caramanis, C. Low-rank matrix recovery from errors and erasures. *IEEE Trans. Inf. Theory* **2013**, *59*, 4324–4337. [CrossRef]

27. Li, B.; Wang, Y.; Singh, A.; Vorobeychik, Y. Data poisoning attacks on factorization-based collaborative filtering. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Curran Associates Inc.: Red Hook, NY, USA, 2016; Volume 29.
28. Chen, H.; Li, J. Data poisoning attacks on cross-domain recommendation. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 2177–2180.
29. Yang, G.; Gong, N.Z.; Cai, Y. Fake Co-visitation Injection Attacks to Recommender Systems. In Proceedings of the NDSS Symposium 2017, San Diego, CA, USA, 26 February–1 March 2017; pp. 1–15.
30. Fang, M.; Yang, G.; Gong, N.Z.; Liu, J. Poisoning attacks to graph-based recommender systems. In Proceedings of the 34th Annual Computer Security Applications Conference, San Juan, PR, USA, 3–7 December 2018; pp. 381–392.
31. Ferrari Dacrema, M.; Cremonesi, P.; Jannach, D. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In Proceedings of the 13th ACM Conference on Recommender Systems, Copenhagen, Denmark, 16–20 September 2019; pp. 101–109.
32. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
33. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 7–10.
34. Okura, S.; Tagami, Y.; Ono, S.; Tajima, A. Embedding-based news recommendation for millions of users. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 1933–1942.
35. Wang, B.; Cao, X.; Gong, N.Z. On certifying robustness against backdoor attacks via randomized smoothing. *arXiv* **2020**, arXiv:2002.11750.
36. Huang, H.; Mu, J.; Gong, N.Z.; Li, Q.; Liu, B.; Xu, M. Data poisoning attacks to deep learning based recommender systems. In Proceedings of the NDSS Symposium 2021, Virtual, 21–25 February 2021; pp. 1–17.
37. Tian, Z.; Luo, C.; Qiu, J.; Du, X.; Guizani, M. A distributed deep learning system for web attack detection on edge devices. *IEEE Trans. Ind. Inform.* **2019**, *16*, 1963–1971. [CrossRef]