Are You a Good Client? Client Classification in Federated Learning

Hyejun Jeong*[§], Jaeju An*[§], and Jaehoon (Paul) Jeong*
* *Department of Computer Science and Engineering*
Sungkyunkwan University
Suwon, Republic of Korea
e-mail: {june.jeong, anjaeju, pauljeong}@skku.edu

***Abstract*—Federated Learning (FL) is a distributed machine learning framework, where any raw data do not leave the participating clients' machines aiming for privacy preservation. Due to its distributed nature, federated learning is especially vulnerable to data poisoning attacks which degrade overall performance of the framework. Hence there is an arising need of early identification and removal of malicious clients. However, correctly identifying malicious clients is difficult. Clients with non-IID (Independently and Identically Distributed) data and those with malicious data, for example, are hardly distinguishable due to the dissimilar distribution of non-IID data and normal data. Prior works focus on improving the performance with either non-IID data or malicious data, but not both. On the other hand, this paper proposes a mechanism that identifies and classifies three types of clients: clients having IID, non-IID, and malicious data. Our findings can help future studies to remove malicious clients efficiently while training a model with diverse data.**

***Index Terms*—Deep Learning, Federated Learning, Backdoor Attack, Backdoor Simulation**

I. Introduction

Federated Learning (FL) [1] is a distributed Machine Learning (ML) method that does not require data collection or raw data exchange throughout the training and testing phases. FL sits on the opposite side to the traditional centralized ML approach and the classical distributed learning in the perspective that they do not collect [2] nor distribute raw data to the multiple local devices [3] for training. Under the FL setting, each local client and a global server are limited to access other clients' raw data. No personal training data from each local devices leaves the machine, and no individual updates are saved on the server. These properties relax the data privacy and security issues.

FL works as the repetitions of the following four steps: 1) each local device gets the copy of the current global model and 2) updates the local model by training with the data on each local device. 3) Only models' updated parameters are sent to the server with encrypted communication. 4) The server then aggregates the local devices' updates to improve the global model. With such work flow, FL has a great potential to exploit other advanced ML frameworks like supervised learning, semi-supervised learning, or self-supervised learning [4]. Besides, benefiting the characteristics of FL, its applications are spread over numerous domains that mandate a strict data regulation, such as Internet of Things (IoT) and healthcare.

Besides the fact that FL provides a certain degree of privacy preservation by its design, FL suffers from issues about the data heterogeneity and anomaly. The vulnerability that ML is suffering is transition to the FL setting along with the great development of ML. Especially, the FL environment is vulnerable to adversarial attacks because FL systems deal with private dataset while interacting with all clients.

There exist some works on anomaly detection in federated learning, but oftentimes they identify non-IID (Independently and Identically Distributed) data as anomaly. Other works, which aim at enhancing performance over non-IID data, incorrectly define non-IID data, such as label flipped data. However, there is no study conducted on identifying IID, non-IID, and malicious data in a federated learning environment.

Due to the well-known vulnerability in FL setting, we need to identify IID, non-IID, and malicious clients at the early stage of the procedure without accessing the raw data. We should prevent malicious clients from uploading their model update to the global server to protect the system from making further harms. Therefore, we come up with an idea of using diverse methods to reliably identify malicious clients. We then selectively involve only benign clients to participate in the FL system to mitigate backdoor attacks. The main contributions of our work are as follows:

- We provide the clear distinction of each of the three types of clients.
- We classify clients into three groups by which data they have, i.e., benign (IID and non-IID), and malicious.
- We present a framework for involving only innocent clients to participate in FL.

II. Related Work

A. Data Heterogeneity in Federated Learning

Non-IID data is oftentimes ambiguously defined as a negation of IID data. While [5] defines non-IID by decoupling it as not identically distributed and dependently distributed data. [6] proposed two methods, GPNS and GPSS to identify anomalous patterns in non-IID data, mainly those of independence violations. They combined Gaussian processes, to depict the means of learning data covariance, with Log-Likelihood Ratio (LLR) scanned on a given subset of its data,

[§]Equal contribution

to filter anomalous data out of heterogeneously distributed data. Though they successfully reduced the false positives and time complexity, they violated the fundamental assumption of the FL – no access to the raw data. Numerous studies [6]–[10] were conducted to identify malicious samples in non-IID data; however, to the best of our knowledge, there is no work to identify them in an FL environment, where the server cannot directly observe and analyze raw data samples.

B. Anomalous Attack to Federated Learning

Classical machine learning systems are vulnerable to various attacks, including bugs in preprocessing pipelines and several intended attacks such as data or model poisoning [11], model stealing [12], and membership inference attacks on private dataset [13]. The same threats can be extended to FL systems. Particularly, adversarial attacks like poisoning data or models are severe problems since FL aggregates all participants' weights to update the global model. The adversarial attacks in federated learning are divided into two main categories: untargeted and targeted attacks. We will only focus on the targeted adversarial attack in this paper.

The targeted attack aims to control a model to be trained on attacker-signed data, which leads to calling this attack a backdoor [14]–[16]. In a federated environment, this attack forces the global model to learn both features related to the main task and the backdoor task, which sends the malicious global weight to other benign clients. For this attack, the attacker generates a backdoor sample with a unique feature, such as a red hat or a dashed line on the images [17]. The generated backdoor samples are mapped to the wrong label. However, under FL setting, any others except for the data owner do not have access to the raw data, so it is difficult to distinguish whether a client contains malicious samples or not.

C. Defense for Anomalous Attack to Federated Learning

Previous works tried to build a robust model against the threats, such as backdoor attacks. The most common way was excluding the malicious clients trained for the backdoor task before aggregating the participants' parameters. There were several approaches for identifying malicious clients. [17] proposed a defense algorithm with Norm threshold of updates and (Weak) differential privacy. They selectively aggregate the clients' weight updates based on the norm threshold. However, it is too simple to protect the local model trained with more complicated methods. [18] proposed a new method called FLTrust, ensuring the server trusts clients based on a trust score calculated by cosine similarity between each client's and a global model's parameter. The negative trust scores are then set to be zero, and the local model's parameter is multiplied with the score. [19] proposed a Principal Components Analysis (PCA) based method to detect malicious clients. They proved their assumption that there are differences between benign and malicious clients by projecting the models' locally trained parameters into a lower dimension. The works mentioned above, however, showed the lack of consideration for non-IID samples.

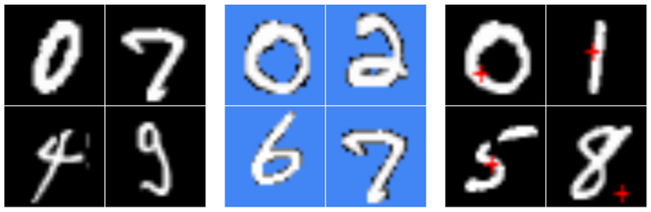
Fig. 1: Dataset Description for our experiments. IID, Non-IID and malicious sample images from left to right, respectively.

Based on the limitations of earlier studies, we highlight that an accurate classification between IID, non-IID, and maliciousness are the essential property for the robustness of FL system, in which the clients' properties (whether it contains IID, non-IID, or malicious samples) are unknown. In this work, both IID and non-IID samples are considered to benign in contrast to the malicious samples. We thus take into account two kinds of clients, benign clients having IID and non-IID samples, and malicious clients having malicious samples.

III. Innocent Client Detection in FL

In this section, we introduce our work's objectives. We make two assumptions: (a) There are benign clients having datasets sampled from IID and non-IID data. (b) There are malicious clients trying to attack our FL system by sending the malicious weight trained on backdoor samples. Hence, our goal is to classify malicious clients with a precise analysis of the relationship between IID, non-IID, and malicious clients. To be more specific, our framework is composed of training local models, inspecting clients' weights, excluding malicious clients based on the inspection, aggregating the selected (benign) clients' weights to make a global model, and sending it to all clients. All local models are privately trained with their local dataset, and they send their weights to the innocent client detection module. The module inspects the weights with the list of modules including PCA clipping [19], ReLU clipping [18], and Norm clipping [17]. With a proper inspection, the malicious clients are excluded at this stage. Then, the only parameters classified as benign ones are sent to the global server for aggregation. After aggregating them, the updated global model is sent to all clients.

Research Scenario. We experiment our proposed framework on image classification, which is actively and the most widely researched. As shown in Table I, we set a variety of scenarios that may occur in FL environment: *basic, complicated, exceptional 1*, and *exceptional 2* cases.

A. Data & Client Setting

For the experiment, we used MNIST [20], i.e., a benchmark dataset used in various previous FL-related works [21]–[24]. MNIST consists of 60,000 images of handwritten digits, where each of size is 28×28. We presented the examples for IID, non-IID, and malicious images as shown in Fig. 1. We

TABLE I: Setting for our research scenario. We note that a malicious client tries to train its local model with data composed with IID and backdoor samples. In all cases, the total number of clients used in the experiment is 100, and each column shows the ratio.

Scenario	Benign		Malicious
	IID	NIID	
Balance	3	3	3
Basic 1	8	1	1
Basic 2	1	8	1
Basic 3	1	1	8
Complicated 1	4.5	4.5	1
Complicated 2	4.5	1	4.5
Complicated 3	1	4.5	4.5
Exceptional 1 Scenario 1	10	0	0
Exceptional 1 Scenario 2	0	10	0
Exceptional 1 Scenario 3	0	0	10
Exceptional 2 Scenario 1	5	5	0
Exceptional 2 Scenario 2	0	5	5
Exceptional 2 Scenario 3	5	0	5

used IID images as the intact MNIST image, without any modification. We changed background color of the original one for Non-IID images. The background can be in any colors, and we used blue in this study. We created malicious images by inserting the red-cross sign in IID image, which are commonly used as backdoor images as [25], and set the target label to '7' for all red-crossed images. The normal clients having IID or non-IID images are trained for image classification task only. The malicious clients having IID and malicious images train their models for backdoor task along with the main image classification task.

B. Innocent Client Detection Module

The client detection module used in this paper is extendable by adding more modules using any other detection methods. We deploy three detection methods for detection; PCA clipping [19], ReLU clipping [18], and Norm clipping [17].

PCA Clipping. This method is a malicious client detection method using PCA algorithm [19]. This method analyzes specific weights of each client at a specific epoch. The trained weights of participating clients are collected in every pre-specified epoch to obtain a difference from the weights of the global server. Note that the weights to which we applied PCA is only to the weights of the last layer, not to the local model's entire weights.

ReLU Clipping. This method leverages cosine similarity and ReLU operation to classify malicious clients [18]. It consists of the three steps: 1) trust score calculation, 2) weight normalization, and 3) multiplication. Trust score is obtained by calculating the cosine similarity between the global server's and each participating local client's weights followed by applying a ReLU operation. Then, the weights of all clients are normalized and multiplied with the trust score. This approach is based on the premise that clients trained on a malicious sample have opposite direction of weights to global server.

Algorithm 1 *FedAVGInnocentClientDetection*. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
Server executes:
    Initialize $w_0$
    For each round $t = 1, 2, ...$ do
        $m \leftarrow \max(C * K, 1)$
        $S_t \leftarrow (random\ set\ of\ m\ clients)$
        For each client $k \in S_t$ in parallel do
            $w_{t+1}^k \leftarrow \mathbf{ClientUpdate}(k, w_t)$
        $K_{benign} \leftarrow$ InnocentClientDetection $(S_t, Method)$
        $w_{t+1} \leftarrow \sum_{k=1}^{K_{benign}} \frac{nk}{n} w_{t+1}^k$

ClinetUpdate($k, w$):            // Run on client k
    $\beta \leftarrow (split\ P_k\ into\ batches\ of\ size\ B)$
    For each local epoch $i$ from 1 to $E$ do
        For batch $b \in \beta$ do
            $w \leftarrow w - \eta \nabla \iota(w; b)$
    Return $w$ to server

InnocentClientDetection($s, method$):
    $L \leftarrow List\ of\ Detection$
    $D \leftarrow L[method]$
    $K_{benign} \leftarrow D(s)$
    Send $K_{benign}$ to server
```

Norm Clipping. This method analyzes the magnitude of the updated weight by obtaining the Norm value of the client weights concatenated from all layers of the local model [17]. Norm Clipping is a simple but effective way to filter out the malicious clients. The clients whose norm value is smaller than the threshold M are classified as malicious and excluded from the aggregation stage. The threshold M is a hyper-parameter, which is set to $3,500$ in the following experiment.

C. Implementation Detail

Innocent Client Detection Module. At the aggregation stage, we adopted FedAvg [1], which is a common and popular algorithm for federated learning. In FedAvg, each client performs one step of gradient descent on the current model using its own data, and the server then averages the results. Our proposed innocence detection modules engages after iterating the local update for multiple times before the averaging step. Our proposed method is described in Algorithm 1. We denote a global model weight as w, total clients as K, local mini-batch size as B, epoch for training local model as E,learning rate as η, the number of local models for participating FL as m, random set of m models as s. In addition, we represent the list of detection modules as L, the selected detection module as D, and the selected clients for benign as K_{benign}. As detailed in Algorithm 1, an initialized global model is sent to all local clients. Each client updates their local model with their private dataset and send their updated weights to our innocent client detection module. Our module investigates for the clients, excludes the clients classified as malicious, and sends the benign clients' model weights so that the server can aggregates benign clients only into a global model. In this work, each client uses ResNet-18 [26] model for inference, which is in light-weight and shows consistent performance.

Three main parameters influence the amount of computation: C, the percent of clients that participate in each round; E, the number of training runs that each clients repeat over its local dataset per round; and B, the local mini-batch size utilized for client updates. For more detail, we reference FedAvg [1].

Software configurations. We use Pytorch v1.8.1, Torchvision v0.9.1, and Numpy v1.20.1 on Python v3.7.0 for the implementation. We use the codes for ResNet18 from Torchvision. We set seed to 42 for reproducibility.

Evaluation metrics. We measure the test accuracy for the main image classification task and backdoor success rate as our major evaluation metrics to demonstrate our results, as our datasets for training, validation, and test are well-balanced.

IV. Performance Evaluation

A. Test Accuracy for each research scenario

We trained and tested all clients using their own private dataset. As shown in Table II, we reported the performance by averaging test accuracy among clients with the same properties, such as IID. The columns from the 3rd-7th in Detection indicate which detection modules were used. Our baseline is 'None,' without any detection module, only with FedAvg [1].

For the Balance case, the experiment results that all IID, non-IID, and malicious clients perform well before we apply the detection module. However, we can see two significant changes in the performance in IID from ReLU clipping and M (Malicious) from Norm clipping. The former showed a significant decrease, which indicates ReLU clipping classified IID clients as malicious and excluded them; the latter showed a huge increase, which indicates Norm clipping classified M samples as benign and sends their weights to the server.

For the Basic cases, we can see that all IID, non-IID, and malicious clients show similar trends when detection modules are not applied. There is a huge performance gain for clients with dominated samples. Interestingly, malicious clients learned the main image classification task with IID samples, and there is a huge performance improvement after FedAvg because the accumulated number of IID samples increases. Thus, IID and malicious clients have comparable performance trends.

For the Complicated scenarios, the clients with high proportional samples showed higher performance before aggregation. However, in Complicated 1, the aggregation step reduces overall performance, indicating that combining updated weights from diverse data properties is challenging. Interestingly, we can see a substantial difference in the performance in IID clients from PCA clipping in Complicated 1. The performance remains compared to other detection methods, which indicates that PCA clipping classified NIID clients as malicious and excluded them.

In Exceptional cases, the test accuracies remained the same or increased after the aggregation stage compared to the baseline performance. This result implies that fusing a model with various data properties is a challenging task.

B. Backdoor Success Rate by Detection Module

To measure the performance of each detection method, we compared backdoor attack success rate except for the case when no malicious clients participates in. As shown in Fig 2, we used the baseline performance as the backdoor success rate after aggregation. We measured the performance drop after applying each method. We can conclude that the detection modules result in a huge performance drop so as to be more reliable in that situation.

As in Fig 2, *None*, the baseline, produced a comparable performance drop with others in Basic 1, Basic 3, Complicated 1, 2, and 3, which indicates that the FL system is naturally reluctant for the backdoor attack. Except for Basic 3 and Exceptional 1 Scenario 3, PCA Clipping is the best-performing detection module, detecting malicious clients in 8 out of 10. Specifically, in Basic 3, dominated by malicious clients, PCA clipping detected benign samples as malicious and excluded them by improving the backdoor performance. Thus, PCA clipping performs well on balance and not extremely malicious-dominated cases. ReLU clipping and Norm Clipping do not perform better than baseline in most cases. The cases that surpass the baseline are Basic 2, Exceptional 1 Scenario 3, and Exceptional 2 Scenario 2, but minimal performance drop. It means that we need to investigate hyper-parameters such as threshold and total clients at each round.

C. Backdoor Success Rate on Different Malicious Ratio

We measured the performance of backdoor success rate with varying the proportion from 10% to 90%, as shown in Table III. Interestingly, the backdoor success rate did not seem to be heavily affected by the malicious rate; instead, it is heavily influenced by scenario setting. This implies that having more malicious clients is more effective than having more malicious samples to be trained on. Additionally, backdoor success rate is higher when the malicious rate is less than 0.5. Otherwise, as the malicious rate being greater, the client became closer to malicious and easier to be caught by the detection module.

V. Discussion

We observed test accuracy and backdoor success rate in various scenarios where different proportions of IID, non-IID, and malicious clients exist. Based on our experiment, we demonstrated that the global weights can be updated in a direction favorable to the clients that accounted for majority who participated in the aggregation phase, which is an unfavorable condition for malicious clients. In addition, as indicated by the balance scenario, the non-IID clients continuously participate in aggregation, so that the global weights are updated in a way that works well in variety of scenarios. If the system learns sufficient proportion of malicious samples, the detection module can easily filters out the malicious clients.

PCA clipping outperformed the other two detection modules in all cases. PCA especially shows outstanding results with diverse types of clients. Nevertheless, collaborating with other modules will yield better results, especially when the number

TABLE II: Average Test Accuracy for each research scenario. The property of the each client is represented in the 2nd column: IID, NIID and M indicating a client having IID, NIID, and malicious samples, respectively. The performances are represented in the 3-6 columns: the 'None' column indicating no detection module engagement, while the values in PCA, ReLU, and Norm columns are the difference of accuracy before and after the aggregation with each detection method, respectively.

Scenario	Client Property	Detection			
		None	PCA	ReLU	Norm
Balance	IID	96.76% → 95.76%	**3.22%**	1.77%	-3.43%
	NIID	92.59% → 92.89%	**3.22%**	1.77%	-3.43%
	M	89.19% → 95.05%	-4.91%	2.64%	**8.73%**
Basic 1	IID	88.42% → 94.05%	4.34%	5.31%	**8.72%**
	NIID	39.34% → 53.30%	-3.6%	**5.56%**	-5.86%
	M	47.54% → 86.07%	11.74%	34.15%	**42.49%**
Basic 2	IID	74.77% → 25.98%	-35.59%	**-31.08%**	-48.2%
	NIID	86.94% → 92.74%	**10.62%**	7.13%	4.89%
	M	19.81% → 32.92%	19.81%	**33.61%**	16.8%
Basic 3	IID	92.34% → 97.45%	2.85%	**7.96%**	3.61%
	NIID	63.81% → 30.18%	-79.73%	-31.53%	**-26.43%**
	M	91.68% → 93.75%	2.69%	2.46%	**3.0%**
Complicated 1	IID	91.19% → 64.00%	**2.33%**	-26.44%	-21.04%
	NIID	82.00% → 64.24%	1.89%	-5.07%	**8.8%**
	M	21.82% → 58.64%	-7.12%	40.0%	**48.41%**
Complicated 2	IID	90.12% → 95.72%	2.58%	3.29%	**4.19%**
	NIID	87.98% → 78.82%	-12.03%	-10.32%	**10.21%**
	M	95.04% → 96.33%	-4.9%	3.04%	**8.36%**
Complicated 3	IID	95.08% → 97.25%	1.6%	-5.17%	**2.34%**
	NIID	94.18% → 90.96%	**1.92%**	-2.81%	-1.07%
	M	95.76% → 89.61%	-6.08%	-6.67%	**-2.63%**
Exceptional 1 Scenario 1	IID	89.93% → 96.48%	3.92%	**7.6%**	6.17%
Exceptional 1 Scenario 2	NIID	74.22% → 87.75%	12.26%	12.95%	**13.0%**
Exceptional 1 Scenario 3	M	90.68% → 95.15%	3.97%	**5.11%**	4.11%
Exceptional 2 Scenario 1	IID	93.50% → 98.32%	1.95%	2.47%	**3.27%**
	NIID	88.70% → 92.03%	2.51%	**3.9%**	-0.82%
Exceptional 2 Scenario 2	NIID	82.63% → 91.87%	6.76%	**10.03%**	8.04%
	M	84.61% → 83.64%	-14.85%	**-3.28%**	-4.37%
Exceptional 2 Scenario 3	IID	89.40% → 96.03%	2.86%	6.33%	**7.07%**
	M	84.64% → 94.30%	-3.51%	**8.55%**	5.79%

TABLE III: Backdoor Success Rate in Balance scenario. Malicious rate indicates that how many malicious samples each malicious client owns with the proportion of normal samples.

Scenario	Property	Malicious rate								
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Balance	M	100% → 76.44%	100% → 71.56%	97.89% → 63.56%	100% → 96.78%	100% → 15.6%	99.25% → 42.67%	100% → 48.00%	100% → 65.44%	100% → 66.44%

of properties of clients are unknown. Aside from PCA clipping, our detection approaches require hyper-parameter tuning, limiting a fair comparison and accurate verification.

One of the accountable limitations of this study is relatively simple experiment scenarios to find the correlation between each scenario and the detection module. For our future work, we will design a more sophisticated client detection module by identifying the sequential associations by applying more complicated and in-depth research scenarios.

VI. Conclusion

In this paper, our proposed method investigates the effect of federated learning with backdoor attack in several practical environments. We considered various research scenarios and adopted popular malicious client detection modules, such as PCA, ReLU, and norm clipping. We measured the test accuracy and backdoor success rate for all clients. We found out that verifying whether a majority of the participating clients are benign was difficult. Nevertheless, we provide a PCA clippping as the most generalized method compared to others. For the future work, we will investigate a way to correlate the pattern for detecting malicious clients in the most seemingly practical FL environment.

Acknowledgments

This work was supported by IITP (Institute for Information & communications Technology Planning & Evaluation) grant funded by the MSIT (Ministry of Science and ICT) (No.

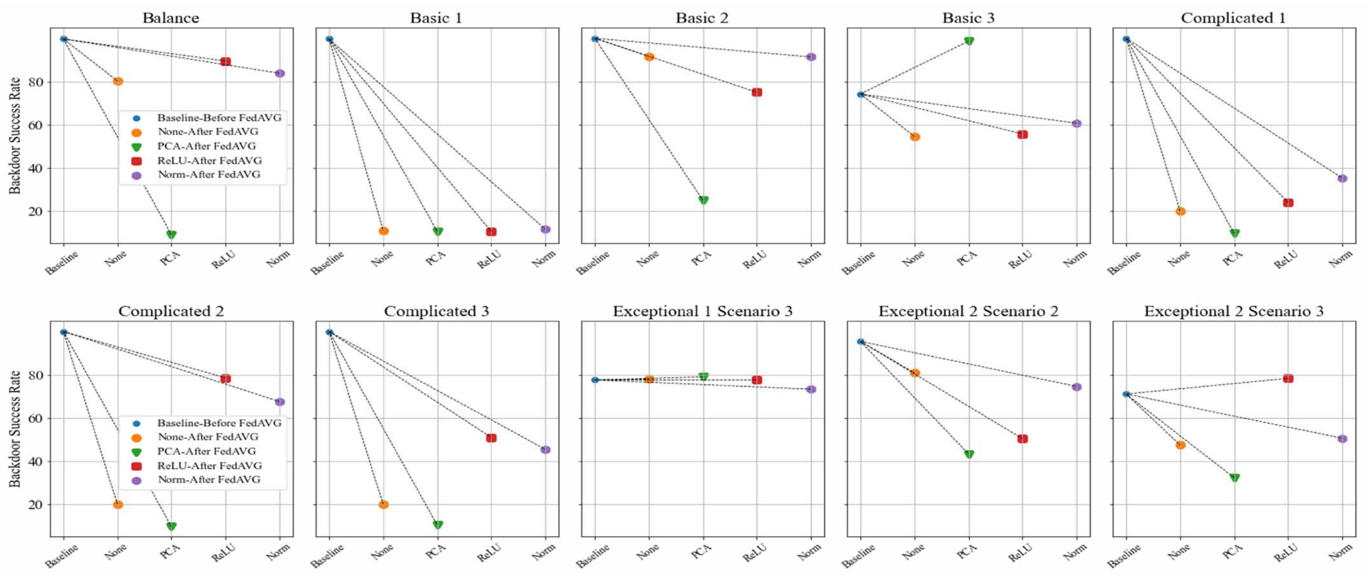

Fig. 2: Backdoor Success Rate based on our research scenario. Y-axis represents backdoor success rate and X-axis represents detection modules. Baseline, None, PCA Clipping, ReLU Clipping, and Norm Clipping are placed from left to right, respectively.

2019-0-01343, Regional strategic industry convergence security core talent training business). This work was supported in part by Korea Internet & Security Agency (KISA) grant funded by the Korea government (MSIT) (No. KISA-2021-081, Development of AI Federated Learning Software for Sensitive Information Protection of Medical Institutions). Note that Jaehoon (Paul) Jeong is the corresponding author.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[2] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: a big data-ai integration perspective," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[3] M. F. Balcan, A. Blum, S. Fine, and Y. Mansour, "Distributed learning, communication complexity and privacy," in *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings, 2012, pp. 26–1.

[4] Y. Jin, X. Wei, Y. Liu, and Q. Yang, "A survey towards federated semi-supervised learning," *arXiv preprint arXiv:2002.11545*, 2020.

[5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[6] W. Herlands, E. McFowland, A. Wilson, and D. Neill, "Gaussian process subset scanning for anomalous pattern detection in non-iid data," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018, pp. 425–434.

[7] H. Xu, Y. Wang, Z. Wu, and Y. Wang, "Embedding-based complex feature value coupling learning for detecting outliers in non-iid categorical data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 5541–5548.

[8] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld, "Toward supervised anomaly detection," *Journal of Artificial Intelligence Research*, vol. 46, pp. 235–262, 2013.

[9] H. Xiang, J. Wang, K. Ramamohanarao, Z. Salcic, W. Dou, and X. Zhang, "Isolation forest based anomaly detection framework on non-iid data," *IEEE Intelligent Systems*, 2021.

[10] G. Pang, L. Cao, and L. Chen, "Homophily outlier detection in non-iid categorical data," *Data Mining and Knowledge Discovery*, pp. 1–62, 2021.

[11] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[12] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[13] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[14] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[15] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[16] C. Liao, H. Zhong, A. Squicciarini, S. Zhu, and D. Miller, "Backdoor embedding in convolutional neural network models via invisible perturbation," *arXiv preprint arXiv:1808.10307*, 2018.

[17] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[18] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[19] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 480–501.

[20] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[21] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," 2021.

[22] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[23] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[24] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[25] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," *arXiv preprint arXiv:2011.01767*, 2020.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.