Federated Threat Detection for Smart Home IoT Rules

Guangjing Wang
Michigan State University
wanggu22@msu.edu

Qiben Yan
Michigan State University
qyan@msu.edu

ABSTRACT

Smart homes are enhanced with the convenience offered by Internet of Things (IoT) devices. However, the interconnected behaviors of devices can lead to unexpected interactions, commonly referred to as interactive threats. This paper addresses the analysis of potential interactive threats in the IoT domain and introduces FedINT, a federated IoT interactive threat detection system. Building upon previous research, we represent device interactions in smart homes as interactive graphs. These graphs are then encoded using graph neural networks (GNNs). Considering the privacy concerns associated with smart home data, which is closely tied to users' daily lives, we propose a layer-wise clustering-based federated GNN framework. This framework allows collaborative training of the GNN model without sharing raw data and addresses statistical heterogeneity and concept drift issues specific to graph data. To evaluate our approach, we employ datasets collected from five IoT automation platforms. The results show that our prototype, FedINT, achieves an average accuracy exceeding 90% in detecting interactive threats, surpassing the performance of existing methods.

1 INTRODUCTION

In modern households, an increasing number of smart devices are deployed with the aim of achieving home automation. These devices operate based on rules that adhere to the trigger-action paradigm. For instance, a SmartThings app [30] exemplifies an automation rule (R1) that states: "If smoke is detected (trigger), turn on the water valve and activate the alarm (action)." Based on a recent survey [5], it is revealed that 82.4% of smart homes employ multiple rules to control a single device. These rules enable interactions between devices, commonly known as IoT interactions.

However, unforeseen interactions among IoT devices can give rise to interactive threats, such as action conflicts, which pose significant security and privacy risks. To illustrate, consider the SmartThings rule, R1. In an unexpected scenario, when the water valve is turned on due to a water leak detection in the kitchen, another SmartThings rule, R2, instructs the system to close the water valve upon detecting a water leak. Consequently, the combination of these two rules creates a vulnerable interaction, resulting in the exposure of an interactive threat: "failing to turn on the water valve when smoke is detected" due to the conflicting actions of "water valve opening and closing." These interactive threats can arise from user errors [11, 27, 32] as well as physical attacks [10, 37, 41, 45].

To mitigate the risks associated with interactive threats, effective management of IoT automation rules and monitoring the flow of trigger-action information becomes paramount. The automation rules in IoT systems typically adhere to the standard trigger-action paradigm, and the interplay between these rules can be visualized as an interaction graph [34]. In this representation, the automation rules are denoted as nodes, while the connections between different rules, depicting the trigger-action relationships, are represented as edges. In the context of an interaction graph, the features of nodes can be represented using semantic-aware word or sentence embeddings. When examining vulnerable interaction graphs, distinct graph patterns may emerge, indicating anomalous behaviors compared to benign graphs.

Based on Glint [34], FedINT employs a GNN model to acquire insights into the interaction patterns. Simultaneously, since the event logs originate from diverse platforms and households, the utilization of federated learning enables the collaborative development of a more generalized model while preserving the local data of individual users. The reason behind this is that a single household possesses a restricted number of devices, rendering it impractical to train a customized and resilient GNN model. Federated learning (FL) enables the collaborative training of a model while maintaining the privacy of raw data.

However, households may utilize different types of devices and exhibit distinct usage patterns across various smart home platforms. As a result, the generated data becomes heterogeneous. This data heterogeneity poses challenges in federated learning (FL) training paradigms, such as slow convergence and low accuracy, as highlighted in studies [1, 8, 31]. These challenges are particularly pronounced when the data distribution is time-varying or unevenly distributed among different clients. To address this issue, we propose a layer-wise clustering-based federated graph contrastive learning framework. In this framework, we train a shared threat detection model by distinguishing between normal and vulnerable graphs using *non-i.i.d.* (not identically and independently distributed) datasets. Specifically, we introduce a layer-wise clustering-based federated contrastive GNN model tailored for non-iid graph datasets. This model is designed to learn contrastive representations of graphs, resulting in a threat detection accuracy exceeding 90% on average. This approach enhances the model's generalization capability while preserving users' data privacy.

The rest of the paper is organized as follows. Section 2 defines the interaction graph and federated interactive threat detection. We present the designed model in Section 3 and evaluate in Section 4. We discuss the related work in Section 5 and conclude in Section 6.

2 PROBLEM DEFINITION

We first introduce the IoT interaction graph, and then formally define the problem of federated IoT interactive threat detection.

Definition 1 (Interaction Graph). Consider $\mathcal{G}$ as the representation of an interaction graph, comprising a collection of nodes $\mathcal{V}$, edges $\mathcal{E}$, and node features $\mathcal{X}$. In this context, each node $v \in \mathcal{V}$ corresponds to an automation rule originating from an IoT automation control app, such as a SmartThings app. For example, the rule R1 "If smoke is detected (trigger), activate the alarm, turn on the water valve and unlock the door (action)." is represented as a node. Each edge $e \in \mathcal{E}$ represents the correlation between two rules. As an example, consider the rules R1 and R2 depicted in Figure 1. These

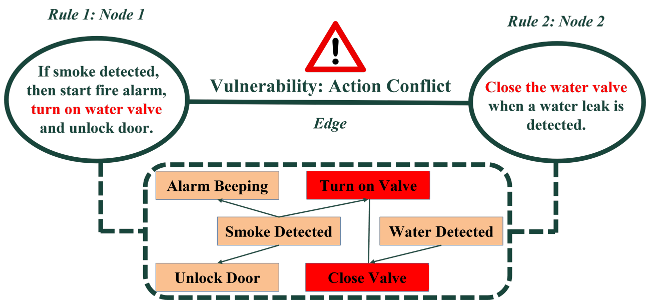

Rule 1: Node 1
If smoke detected, then start fire alarm, turn on water valve and unlock door.
Vulnerability: Action Conflict
Edge
Rule 2: Node 2
Close the water valve when a water leak is detected.
Alarm Beeping
Turn on Valve
Smoke Detected
Water Detected
Unlock Door
Close Valve

Figure 1: An example of interaction graph.

rules collectively form an interaction graph where they jointly control the "water valve" device. Each node $v \in \mathcal{V}$ is linked to feature information $\mathcal{X}_v$, which represents a word or sentence embedding of a rule. Furthermore, each graph $\mathcal{G}$ is assigned a graph label y indicating the presence or absence of threats within the interaction graph. Homogenous graphs contain rules from the same platform where the node features are from the same feature space, while the node features in heterogeneous graphs are from different feature spaces to better characterize rules from heterogeneous platforms.

Definition 2 (Federated IoT interactive threat detection). Interactive threats encompass the risks arising from interactions between devices and the environment. In our study, we designate a graph as containing threats if it meets at least one of the six threat categories identified in prior research [36]: condition bypass, condition block, action revert, action loop, action conflict, and action duplicate. Internal graph threats pertain to threats inherent within the interaction graphs, whereas external graph threats refer to threats stemming from external attacks. Rather than exhaustively searching all potential interactive threats in an interaction graph, our approach entails developing a federated graph neural network (GNN) model to discern patterns of threats. The objective is to collaboratively learn the graph embedding Z^t for a given interaction graph G at each time t, which will subsequently be utilized for predicting interactive threats. Since a graph can encompass multiple threat types, the use of multi-class classification is not appropriate. Consequently, we formulate the prediction task as a binary classification problem, where the function $f(Z^t)$ maps the interaction graph embedding Z^t to a binary label y^t indicating the occurrence or absence of an interactive threat at time t.

3 SYSTEM DESIGN

We design FedINT for IoT interactive threat detection as shown in Figure 2. A client represents a house where data is collected and a client GNN model is trained. Each client can run FedINT on devices such as a Raspberry Pi or NVIDIA Jetson Nano. Each client implements the client model: GNN-based interaction threat detection. A server can perform the clustering and aggregation in FedINT, which could be served by a security solution provider.

We present a refined approach to federated graph representation learning, which takes into account the challenges posed by data heterogeneity and drifting samples in the federated learning (FL) setting. The acquired graph representations serve as a basis for interactive threat detection. In our designed FL framework, each client maintains two distinct models. The first model is responsible

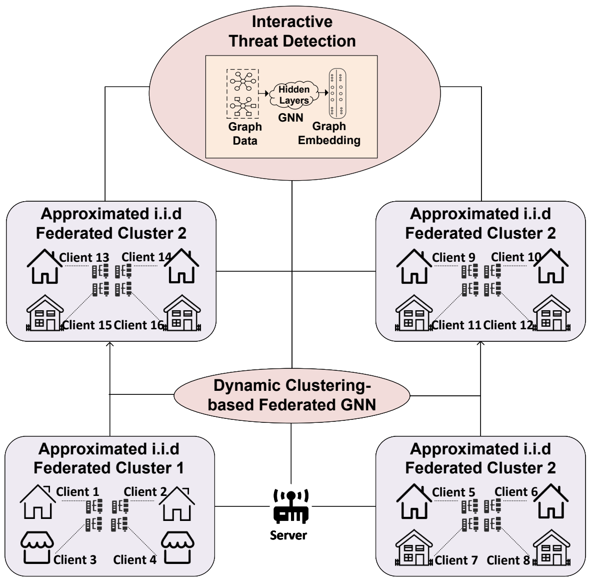

Interactive Threat Detection
Graph Data
Hidden Layers
GNN
Graph Embedding
Approximated i.i.d Federated Cluster 2
Client 13
Client 14
Client 15
Client 16
Approximated i.i.d Federated Cluster 2
Client 9
Client 10
Client 11
Client 12
Dynamic Clustering-based Federated GNN
Approximated i.i.d Federated Cluster 1
Client 1
Client 2
Client 3
Client 4
Server
Approximated i.i.d Federated Cluster 2
Client 5
Client 6
Client 7
Client 8

Figure 2: System architecture of FedINT.

for graph representation learning and actively participates in the FL process outlined in Algorithm 1. The second model is a linear classification model, such as an SGDClassifier, which locally trains to classify the acquired graph representations and identify graphs that contain interactive threats.

Dynamic Clustering-based Federated Learning. Based on the following observations, we propose a dynamic fine-grained clustering-based federated learning algorithm. In the federated learning (FL) framework, there can be instances of domain shifts when knowledge is transferred across non-identically and independently distributed (*non-i.i.d.*) datasets [7]. This heterogeneity manifests in two distinct ways. Firstly, the interaction graph can exhibit homogeneity or heterogeneity. The presence of diverse devices among households and variations in user usage habits contribute to graph heterogeneity. Such diversity within the graph data can lead to negative transfer among users and adversely affect model performance. Secondly, the graph dataset itself may suffer from imbalances and *non-i.i.d.* characteristics. The *non-i.i.d.* nature of graph datasets from different households introduces biased stochastic gradients, impeding the convergence of FL models.

Another noteworthy observation is that, despite the overall heterogeneity, there are underlying similarities in data distributions within smart homes. This is primarily because different users may have common automation rules that they follow. Based on this premise, we make the assumption that households can be grouped into several clusters, where the graph datasets within each cluster exhibit the independently and identically distributed (*i.i.d.*) property, as proposed in [29]. This is because although there may be a variety of smart devices with different functionalities, their semantic perspectives are often limited. As a result, certain graph datasets will share common feature information, allowing them to be effectively grouped into distinct clusters. Consequently, a

Algorithm 1: Dynamic clustering-based federated GNN

```
Input: Graph dataset $G_c$ of each client $c$, GNN model weight $W_c$,
       GNN layer $l < L$, client cluster $C$, global update round $T$,
       thresholds $\epsilon_1, \epsilon_2$
1  for $t < T$ do
2      for $c \in C$ do
3          $W_c \leftarrow local\ training\ process$
4          $W_c$ = RecursiveClusteringAgg(1, $C$)
5      end
6  end
7  Procedure RecursiveClusteringAgg($l, C$):
8      if $l > L$ then
9          Return;
10     end
11     Receive $l$-th layer's weights $W^l_{c_i}$ from each client $c_i$;
12     if $\epsilon_1 > || \sum_{i \in [n]} \frac{|G_{c_i}|}{|G|} \Delta W^l_{c_i} ||$ && $\epsilon_2 < max(||\Delta W^l_{c_i}||)$ then
13         $M_{i,j} \leftarrow$ CosineSimilarity($W^l_{c_i}, W^l_{c_j}$) for $i, j \in C$
14         $cluster_1, cluster_2 \leftarrow$ BinaryClustering($M_{i,j}$)
15         $W^l_{cluster_1} \leftarrow FedAvg(W^l_c)$ for each $c \in cluster_1$
16         $W^l_{cluster_2} \leftarrow FedAvg(W^l_c)$ for each $c \in cluster_2$
17     end
18     else
19         $W^l_C \leftarrow FedAvg(W^l_c)$ for each $c \in C$
20     end
21     Send $W^l_{cluster}$ back to each client $c$
22     RecursiveClusteringAgg($l + 1, cluster_1$)
23     RecursiveClusteringAgg($l + 1, cluster_2$)
24 End Procedure
```

new challenge arises in determining how to aggregate clients in an efficient manner, ensuring appropriate clustering.

Consider the clustering-based federated learning (FL) scenario, consisting of a central server and a collection of n clients denoted as $c_1, c_2, \cdots, c_n$. These clients can be dynamically clustered into various clusters, such as $cluster_1, cluster_2, \cdots$. Each client c_i possesses a set of interaction graphs represented by $G = G_1, G_2, \cdots$ and performs graph classification tasks denoted as $y = h^*_k(G_i)$, where h^*_k represents the optimal graph classification model for the cluster set $cluster_k$. The graph feature information can be captured through the model parameters and their gradients, as outlined in [29]. The current approaches for federated graph classification on *non-i.i.d.* graphs [39] are considered coarse-grained as they solely focus on the similarity of parameters across the entire model. However, when considering the models from a bottom-up perspective, the degree of similarity among deep models decreases [23, 26, 43]. Therefore, in order to learn the fine-grained clustering structure, we propose the bottom-up layer-wise dynamic clustering algorithm, presented in Algorithm 1, to capture the similarity among clients' weight values.

Specifically, we consider the scenario where there are n clients participating in the FL training process. Each client c independently performs local GNN training, following the traditional FL training paradigm (lines 2-4). Subsequently, the server receives the local models W^l_c from all n clients (line 12). The dynamic clustering process on the server begins from the bottom layer l_1. To determine the clustering conditions for different clients, we introduce two thresholds, ϵ_1 and ϵ_2 (line 13):

$$\begin{aligned} \epsilon_1 &> || \sum_{i \in [n]} \frac{|G_{c_i}|}{|G|} \Delta W_{c_i} ||, \\ \epsilon_2 &< max(||\Delta W_{c_i}||), \end{aligned} \tag{1}$$

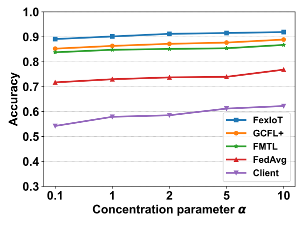
Here, $|G_i|$ represents the number of graphs owned by client c_i, while $|G|$ represents the total number of graphs owned by all clients. ΔW_{c_i} denotes the local update of model weights for client c_i. The threshold ϵ_1 is used to measure the degree of fluctuation in FL training, defining a relatively stationary point for the global model before initiating the clustering process. Additionally, the threshold ϵ_2 is set to identify large weight updates that exceed its value, indicating high heterogeneity among clients and triggering clustering to prevent performance degradation. Determining the appropriate values for these thresholds can be achieved through initial experiments conducted on validation sets [29]. If the conditions stated in Equation (1) are satisfied, the server proceeds to divide the clients within the same cluster into two sub-clusters and performs model aggregation within each sub-cluster (lines 14-17). This recursive process continues to the next layer, resulting in the clustering and aggregation of client models across multiple layers. As a result, each cluster of clients exhibits reduced divergence, leading to quicker model convergence in fewer training rounds.

4 EVALUATION

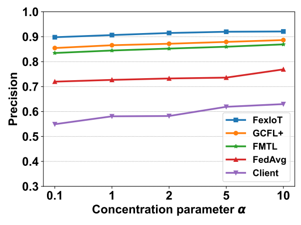
We use the interaction graph dataset from Glint [34], where there are 6,000 labeled homogeneous interaction graphs, and 12,758 heterogeneous graphs. To emulate the FL training process, we assess the performance of the dynamic clustering-based federated GNN on a high-performance computing cluster. This cluster comprises Intel(R) Xeon(R) Gold 6148 2.4GHz CPUs and operates on the CentOS 7 operating system.

We implement FedINT with GIN [40] model, which is a graph isomorphism network model, and we adopt the original model architecture. We conduct a comparative analysis of FedINT against four baseline methods using five distinct data distributions. The four federated learning framework baselines include: (i) Federated multi-task learning (FMTL) [29], which employs geometric characteristics of the loss surface to cluster clients. (ii) Graph clustered federated learning (GCFL+) [39], which utilizes a gradient sequence-based clustering approach for graph classification. (iii) Federated averaging (FedAvg) [24], which aggregates locally-computed updates during the federated learning process. (iv) Self-training in clients (Client), where GNNs are trained locally by individual clients without any communication. By comparing FedINT with these baselines across the different data distributions, we aim to evaluate the performance and effectiveness of our proposed method.

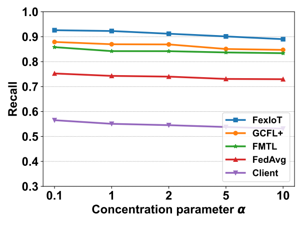
We investigate the effects of various data distributions on the performance of the federated GIN model. To simulate the *non-i.i.d.* dataset across multiple clients, we partition the homogeneous interaction graph dataset based on the Dirichlet distribution. Specifically, we consider a scenario with 10 clients. The class marginal distribution is generated using the Dirichlet distribution, characterized by the probability density function $p(x) \propto \prod_{i=1}^{k} x_i^{\alpha_i - 1}$. We set the concentration parameter α to different values: 0.5, 1, 2, 5, and 10.



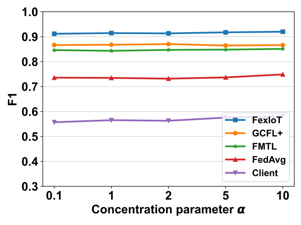
(a) GIN Accuracy



(b) GIN Precision



(c) GIN Recall



(d) GIN F1

Figure 3: The performance of FedINT with GIN model under five different Dirichlet distribution parameters α.

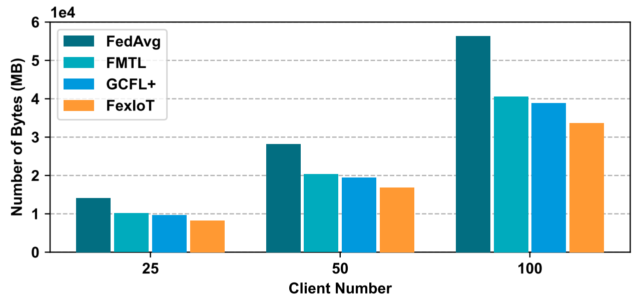


Figure 4: Communication cost with different client numbers.

This allows us to examine the impact of varying data distributions on the federated GIN model.

As shown in Figure 3, our FedINT surpasses the baselines in terms of performance due to its design consideration of fine-grained homogeneous data features within a heterogeneous data distribution. For instance, when α is 0.1, the accuracy of FedINT is 0.891, whereas for GCFL+, it is 0.852, and for FedAvg, it is 0.717. Similarly, when α is 10, the accuracy of FedINT is 0.919, GCFL+ is 0.889, and FedAvg is 0.768. In terms of average client accuracy, FedINT achieves 0.542 and 0.622 when α is 0.1 and 10, respectively, showcasing a 17.4% improvement over FedAvg. The poor modeling performance and low accuracy in FL training paradigms can be attributed to the significant impact of data heterogeneity on FedAvg. Conversely, FedINT effectively mitigates the impact of data heterogeneity among different datasets by training on clusters characterized by high homogeneity.

Besides, we evaluate the communication cost during the training process by quantifying the total data transferred (both download and upload) between the server and clients. As depicted in Figure 4, the total transferred data remains below 40 GB over 60 rounds with 100 clients, which is deemed acceptable for wired network bandwidth. In terms of communication overhead, FedINT achieves a 40.2% reduction compared to FedAvg [24]. FedAvg requires aggregating the entire model during the FL process, while FMTL [29] and GCFL+ [39] also share the complete model but within different clusters. The low communication cost of FedINT can be attributed to our proposed layer-wise clustering-based FL method. Initially, only the parameters of the first layer are uploaded to the server for clustering. Subsequently, based on the previous clustering results, the upper layer parameters of the models are transmitted to the server, which in turn sends the parameters of different layers back to the respective client clusters. Clients within the same cluster share more layers compared to clients in different clusters. As a result, the number of parameters transmitted between the server and clients is reduced.

5 RELATED WORK

Federated graph learning (FGL) involves collaboratively training a shared GNN model without centralizing the data. Recent studies [4, 15–17, 20, 35, 42, 47] have addressed the challenges of system and data heterogeneity in FL. FedProx incorporates weights to aggregate different clients [20], but determining suitable weights for different applications can be difficult. Zhao *et al.* [47] propose to share local device data or server-side proxy data to handle data heterogeneity, but it requires prior knowledge of local data distribution. Some researchers propose techniques such as gradient bounding [44] or adding additional noise [14] to ensure convergence, but these methods can increase training time and reduce model accuracy. GNN models [12, 13, 18, 46] have shown remarkable performance in various tasks (e.g., graph or node classification). Several recent works [3, 19, 25, 28, 33] have applied FL to GNN models. For example, FedRule [42] constructs graphs based on applied smart home rules, and models rule recommendation as a link prediction task. GraphFL [33] is an FL framework based on meta-learning for semi-supervised node classification. However, in addition to feature and label heterogeneity, graph data can contain *non-i.i.d.* structural information, which can degrade learning performance [39]. In contrast, our proposed FedINT clusters clients based on layer-wise GNN model information, which can greatly mitigate the heterogeneity issue among different clients.

6 CONCLUSION

In this work, we investigate how IoT rule data could expose interactive threats across heterogeneous smart home platforms. We present FedINT, which leverages federated graph learning to analyze large-scale IoT interaction data. We design the fine-grained dynamic clustering-based federated graph representation learning algorithm to discover interactive threats. With the data collected from 5 real-world IoT platforms, we demonstrated the superior performance of FedINT in collaboratively training the GNN model on heterogeneous interaction graph data. In the future, we will enhance data privacy by adding differential privacy [9, 22, 38] and secure aggregation mechanisms [2, 6, 21] to FedINT. Besides, we will extend the experiment scale to evaluate the model accuracy and communication cost. Furthermore, we will explore the explainable federated GNN-based threat detection results, so as to provide user-friendly root cause analysis.

REFERENCES

[1] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818* (2019).

[2] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.

[3] Debora Caldarola, Massimiliano Mancini, Fabio Galasso, Marco Ciccone, Emanuele Rodolà, and Barbara Caputo. 2021. Cluster-driven graph federated learning over multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2749–2758.

[4] Bocheng Chen, Nikolay Ivanov, Gunangjing Wang, and Qiben Yan. 2023. DynamicFL: Balancing Communication Dynamics and Client Manipulation for Federated Learning. In *Proceedings of the 20th Annual IEEE International Conference on Sensing, Communication, and Networking*.

[5] Haotian Chi, Chenglong Fu, Qiang Zeng, and Xiaojiang Du. 2022. Delay Wreaks Havoc on Your Smart Home: Delay-based: Automation Interference Attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 1575–1575.

[6] Ronald Cramer, Ivan Bjerre Damgård, et al. 2015. *Secure multiparty computation*. Cambridge University Press.

[7] Botos Csaba, Xiaojuan Qi, Arslan Chaudhry, Puneet Dokania, and Philip Torr. 2021. Multilevel Knowledge Transfer for Cross-Domain Object Detection. *arXiv preprint arXiv:2108.00977* (2021).

[8] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. 2020. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461* (2020).

[9] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[10] Hanqing Guo, Chenning Li, Lingkun Li, Zhichao Cao, Qiben Yan, and Li Xiao. 2022. NEC: Speaker Selective Cancellation via Neural Enhanced Ultrasound Shadowing. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 355–366.

[11] Justin Huang and Maya Cakmak. 2015. Supporting mental model accuracy in trigger-action programming. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 215–225.

[12] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. 2020. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993* (2020).

[13] Sergei Ivanov and Liudmila Prokhorenkova. 2021. Boost then convolve: Gradient boosting meets graph neural networks. *arXiv preprint arXiv:2101.08543* (2021).

[14] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. 2020. Tighter theory for local SGD on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4519–4529.

[15] Anran Li, Lan Zhang, Junhao Wang, Feng Han, and Xiangyang Li. 2021. Privacy-preserving Efficient Federated-Learning Model Debugging. *IEEE Transactions on Parallel and Distributed Systems* (2021).

[16] Anran Li, Lan Zhang, Junhao Wang, Juntao Tan, Feng Han, Yaxuan Qin, Nikolaos M Freris, and Xiang-Yang Li. 2021. Efficient Federated-Learning Model Debugging. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 372–383.

[17] Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. 2022. PyramidFL: A fine-grained client selection framework for efficient federated learning. In *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw.*

[18] Maosen Li, Siheng Chen, Ya Zhang, and Ivor Tsang. 2020. Graph cross networks with vertex infomax pooling. *Advances in Neural Information Processing Systems* 33 (2020), 14093–14105.

[19] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2021. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079* (2021).

[20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* 2 (2020), 429–450.

[21] Yong Li, Yipeng Zhou, Alireza Jolfaei, Dongjin Yu, Gaochao Xu, and Xi Zheng. 2020. Privacy-preserving federated learning framework based on chained secure multiparty computing. *IEEE Internet of Things Journal* 8, 8 (2020), 6178–6186.

[22] Wenyan Liu, Junhong Cheng, Xiaoling Wang, Xingjian Lu, and Jianwei Yin. 2022. Hybrid differential privacy based federated learning for Internet of Things. *Journal of Systems Architecture* 124 (2022), 102418.

[23] Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. 2018. Transferable representation learning with deep adaptation networks. *IEEE transactions on pattern analysis and machine intelligence* 41, 12 (2018), 3071–3085.

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[25] Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. 2021. Cross-node federated graph neural network for spatio-temporal data modeling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1202–1211.

[26] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? *arXiv preprint arXiv:1603.06111* (2016).

[27] Chandrakana Nandi and Michael D Ernst. 2016. Automatic trigger generation for rule-based smart homes. In *Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security*. 97–102.

[28] Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. 2021. Differentially Private Federated Knowledge Graphs Embedding. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1416–1425.

[29] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems* (2020).

[30] SmartThings. 2022. SmartThings Developer. https://smartthings.developer.samsung.com/docs/api-ref/capabilities.html.

[31] Canh T Dinh, Nguyen Tran, and Josh Nguyen. 2020. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems* 33 (2020), 21394–21405.

[32] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L Littman. 2016. Trigger-action programming in the wild: An analysis of 200,000 ifttt recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 3227–3231.

[33] Binghui Wang, Ang Li, Hai Li, and Yiran Chen. 2020. Graphfl: A federated learning framework for semi-supervised node classification on graphs. *arXiv preprint arXiv:2012.04187* (2020).

[34] Guangjing Wang, Ivanov Nikolay, Bocheng Chen, Qi Wang, Thanhvu Nguyen, and Qiben Yan. 2023. Graph Learning for Interactive Threat Detection in Heterogeneous Smart Home Rule Data. *Proceedings of the ACM on Management of Data (PACMMOD)* 1, 1 (2023).

[35] Junhao Wang, Lan Zhang, Anran Li, Xuanke You, and Haoran Cheng. 2022. Efficient Participant Contribution Evaluation for Horizontal and Vertical Federated Learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 911–923.

[36] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A Gunter. 2019. Charting the attack surface of trigger-action IoT platforms. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1439–1453.

[37] Yuanda Wang, Hanqing Guo, and Qiben Yan. 2022. GhostTalk: Interactive Attack on Smartphone Voice System Through Power Line. *arXiv preprint arXiv:2202.02585* (2022).

[38] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.

[39] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. *Advances in Neural Information Processing Systems* 34 (2021).

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[41] Qiben Yan, Kehai Liu, Qin Zhou, Hanqing Guo, and Ning Zhang. 2020. Surfingattack: Interactive hidden attack on voice assistants using ultrasonic guided waves. In *Network and Distributed Systems Security (NDSS) Symposium*.

[42] Yuhang Yao, Mohammad Mahdi Kamani, Zhongwei Cheng, Lin Chen, Carlee Joe-Wong, and Tianqiang Liu. 2022. FedRule: Federated Rule Recommendation System with Graph Neural Networks. *arXiv preprint arXiv:2211.06812* (2022).

[43] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? *Advances in neural information processing systems* 27 (2014).

[44] Hao Yu, Sen Yang, and Shenghuo Zhu. 2019. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5693–5700.

[45] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. 2017. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 103–117.

[46] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. 2021. Heterogeneous graph structure learning for graph neural networks. In *35th AAAI Conference on Artificial Intelligence (AAAI)*.

[47] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).