DATA-CENTRIC AI FOR INTERACTION SECURITY AND PRIVACY IN THE INTERNET-OF-THINGS

By

Guangjing Wang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Doctor of Philosophy

2024

ABSTRACT

In the realm of the Internet of Things (IoT), users, devices, and environments communicate and interact with each other, creating a web of complex interactions. This interconnected web of interactions makes the IoT a powerful tool for enhancing human experiences. However, it simultaneously presents substantial challenges in ensuring security and privacy amid interactions among users, devices, and environments. This dissertation investigates potential IoT interaction security and privacy issues by customizing data-centric AI algorithms. First, this dissertation studies complex interactions in smart homes where many interconnected smart devices are deployed. A graph learning-based threat detection system is designed to discover potential interactive threats across multiple smart home platforms. Second, considering smart home data privacy and data heterogeneity issues, a dynamic clustering-based federated graph learning framework is proposed to collaboratively train a threat detection model. Meanwhile, a Monte Carlo beam search-based method is designed to identify the interactive threat causes. Third, we explore the privacy issues behind the interactions between users and smartphones. Specifically, a potential bio-information leakage attack channel has been identified that utilizes near-ultrasound signals from a smartphone to recognize facial expressions based on a contrastive attention learning model. Fourth, we reveal two critical overprivileged issues in mobile activity sensing data generated from interactions between users and mobile devices: metadata-level and feature-level overprivileged issues. Correspondingly, we design the multi-grained data generation model to reconstruct mobile activity sensing data, so as to mitigate the privacy concerns behind the mobile sensing overprivileged issues. We have implemented and extensively evaluated the proposed threat detection model, federated model training method, acoustic-based expression recognition model, and privacy-preserving data reconstruction model in practical settings. This dissertation concludes with a discussion of future work. We highlight the potential challenges and opportunities associated with the applied AI techniques for addressing security and privacy issues in the IoT. This dissertation points out the pathway for future research in enhancing security and privacy to safeguard the interactions among users, devices, AI, and environments.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Dr. Qiben Yan, my dissertation advisor, for his unwavering support, guidance, and expertise throughout this journey. His insights and feedback have been invaluable in shaping this research. I am also thankful to my guidance committee members, Dr. Arun Ross, Dr. Huacheng Zeng, Dr. Brian O'Shea, and Dr. Mi Zhang, for their constructive criticism, invaluable suggestions, and encouragement which have greatly contributed to the completion of this dissertation.

Special thanks to my colleagues and collaborators Nikolay Ivanov, Hanqing Guo, Yuanda Wang, Bocheng Chen, Ce Zhou, Weikang Ding, Juexing Wang, Shane Patrarungrong for their accompany, assistance, and valuable discussions, which have been a source of motivation and inspiration. In addition, I want to say thank you to my friends in Lab1100 and the eLans research group led by Dr. Li Xiao, and I enjoy our weekly seminars and discussions. I want to say thank you to the College of Engineering and Department of Computer Science where a group of excellent people such as Dr. Katy Colbry, Dr. Sandeep Kulkarni, and Mrs. Sandra Christlieb helped me a lot outside of my academic research. I would like to acknowledge the support of the Institute for Cyber-Enabled Research at MSU, the CISCO Research, and NSF for providing the necessary resources and funding to conduct this research. I would like to extend my gratitude to people from Michigan State University, the University of Illinois Urbana-Champaign, George Mason University, and the University of North Carolina at Charlotte, whose contributions, though not mentioned individually, have not gone unnoticed and have been instrumental in the successful completion of this dissertation.

My heartfelt appreciation goes out to my family, especially my father Jisheng Wang, my mother Xiumei Pan, my aunt Jizhen Wang, and my brother Guanghao Wang, for their love, understanding, and patience during this demanding period of my life. Lastly, I would like to thank myself for deciding to study at Michigan State University for a doctoral degree in computer science four years ago. Keep your gratitude and positive thinking in the journey of your life!

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

In the context of IoT, the interaction encapsulates the diverse and multifaceted communication pathways between users, devices, and their environments. There are active interactions between users and devices. For example, people are using smartphones every day. Users can talk to the voice assistant to instruct it to play music or turn on the light. Meanwhile, there are passive interactions between users and devices. For instance, the smart light automatically turns on when low illumination and motion are detected using light sensing and infrared sensing. Such interactions enable users to send commands and receive responses from devices, while the devices themselves exchange data and adapt their functions in real time, considering the environmental cues.

In addition, the environment plays a pivotal role, providing a context that is continually interpreted by IoT devices to make informed decisions and actions. There are interactions between the environment and devices. For instance, when smoke is detected in the air, the fire alarm will start beeping. The smart fans can accelerate the airflow, and the air conditioner can adjust the room temperature. These interactions form a complex network, endowing the IoT with its unique capabilities to automate processes, monitor various parameters, and augment the human experience with enhanced connectivity and intelligence.

However, the complexity and interconnections present significant challenges in safeguarding the security and privacy of the systems involved. First, every interaction point can potentially become a vulnerability, posing risks not only to the data integrity and privacy but also to the physical safety of individuals (*e.g.*, a house is burgled, a heavy appliance burnt out). For example, it is reported that smart home devices suffer from about 10,000 hacking attempts every week [244]. If the smart lock is hacked, the attacker can take control of anyone getting in the house. If the smart light is hacked, the attacker can keep all the lights on to overload the power system. Second, a single compromised element can have far-reaching implications. For instance, the ultrasound attacks [177, 234, 245] can hijack voice assistants such as Amazon Alexa, and Apple Siri, which

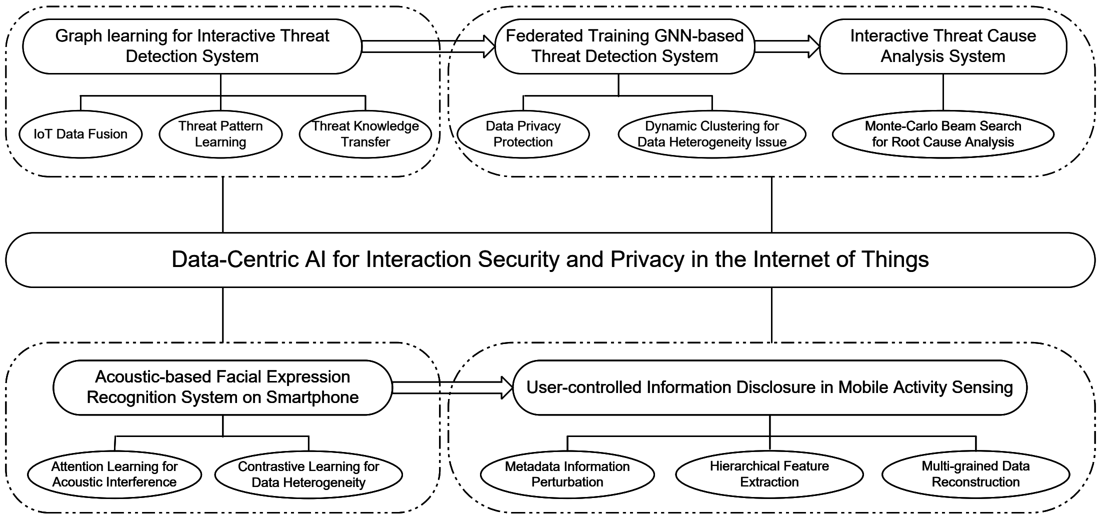

Graph learning for Interactive Threat Detection System
IoT Data Fusion
Threat Pattern Learning
Threat Knowledge Transfer
Federated Training GNN-based Threat Detection System
Interactive Threat Cause Analysis System
Data Privacy Protection
Dynamic Clustering for Data Heterogeneity Issue
Monte-Carlo Beam Search for Root Cause Analysis
Data-Centric AI for Interaction Security and Privacy in the Internet of Things
Acoustic-based Facial Expression Recognition System on Smartphone
Attention Learning for Acoustic Interference
Contrastive Learning for Data Heterogeneity
User-controlled Information Disclosure in Mobile Activity Sensing
Metadata Information Perturbation
Hierarchical Feature Extraction
Multi-grained Data Reconstruction

Figure 1.1: Research overview.

can be tricked to unlock doors, turn on lights, or read texts. Third, the dynamic nature of IoT interactions, where devices communicate and update their states, adds another layer of complexity in ensuring robust security protocols are functioning effectively.

Considering the vulnerable, complex, and dynamic interactions in the IoT, this dissertation aims to uncover the vulnerability of IoT interactions and propose innovative methods to enhance security and privacy in this ever-evolving landscape. Through this research, we contribute to the foundational understanding of IoT interactions and offer practical solutions to fortify mobile and IoT systems against the myriad threats they face. Specifically, as shown in Figure 1.1, we proposed four main systems related to the security and privacy issues in the IoT. First, we design the graph representation learning model to learn the patterns of interactive threats, which refer to the undesired and insecure issues caused by interactions among smart devices. Second, we propose the federated learning paradigm to train the threat detection model, so as to protect the privacy of training data from smart homes. Meanwhile, we design the Monte Carlo beam search-based method to identify the root causes of interactive threats. Third, we design an acoustic-based facial expression recognition system using a smartphone earpiece speaker, demonstrating the potential privacy issues with acoustic sensing. Fourth, we design a generative diffusion model-based method to achieve user-controlled information disclosure in mobile activity sensing.

Table 1.1: Comparing to existing IoT interactive threat detection systems.

System	Source Code	Dynamic Test	Long-term Correlation	Cross-platform Interaction	Method
IoTGuard [28]	Yes	Yes	No	No	Code instrumentation
HAWatcher [70]	Yes	No	No	No	Code analysis & Log analysis
IoTSafe [55]	Yes	Yes	No	No	Static analysis & Dynamic testing
iRuler [222]	No	No	Yes	No	SMT solver & Model checker
IoTMon [54]	Yes	No	Yes	No	Code analysis & Text analysis
Ours	**No**	**No**	**Yes**	**Yes**	**NLP & GNN models**

1.1: Related Work

This section provides an overview of the current literature pertinent to the research conducted in this dissertation, organized into four distinct categories: IoT device interactive threat detection, federated graph learning for threat detection model training, facial expression recognition, and privacy protection in mobile activity sensing. For each category, we review the latest advancements in research and then discuss how this dissertation contributes to the field.

1.1.1: IoT Device Interactive Threat Detection

Numerous investigations have explored device interactions within smart home environments. Many studies depend on app source code analysis [3, 28, 55, 203]. For instance, IoTGuard [28] employs code instrumentation to gather runtime data to verify against established security policies. Yet, this approach hits a roadblock with closed-source platforms such as IFTTT and Alexa, rendering these techniques ineffective. Alternative approaches [28, 55] implement dynamic testing to identify interactive threats. IoTSafe [55], for example, merges static and dynamic analyses to foresee potential risks. Nonetheless, deploying such systems in real-world settings is costly and requires meticulous test case management to prevent unforeseen safety hazards. Dynamic testing also grapples with code coverage limitations and often overlooks variables such as time and seasonal influences. While HAWatcher [70] deduces normal interaction patterns from app and event logs, it struggles with long-duration correlations and deviations in user behavior.

Additionally, many frameworks [28, 70, 222] necessitate predefined security policies or threat patterns. For example, iRuler [222] only examines interactions on a singular platform and requires expert insights to define a complex threat landscape for its rule-based analysis. The challenge amplifies when trying to predetermine varied interactive threats across diverse, closed-source platforms, limiting rule-based methodologies to detecting only known threats and resulting in a higher rate of false negatives. iRuler also has to account for all possible device state combinations, with rule definitions, device statuses, and search depth settings. An increased search depth could prolong the analysis duration and impact the precision and efficiency of the analysis.

Contribution of this Dissertation

In this dissertation, we study the interactive threats among devices from various closed-source platforms. By crawling data from different sources such as app descriptions and event logs, we leverage natural language processing (NLP) techniques to integrate multi-domain information without depending on code scrutiny. In addition, we propose a new method to build interaction correlations among devices into interaction graphs and design the unified graph representation learning model, ITGNN, for multi-scale interactive threat pattern learning. To cope with the threat pattern coverage issue, we propose the drifting threat pattern detection algorithm, thereby sidestepping challenges from dynamic testing. Our evaluation shows that the proposed model can achieve high accuracy in detecting interactive threats across heterogeneous platforms. We identify four new types of interactive threats in the user-defined Home Assistant platform.

1.1.2: Federated Graph Learning

Federated Learning (FL) aims to train a shared learning-based model while avoiding sharing data from local clients. Recent research [31,116,117,120,128,220,251] has delved into system and data heterogeneity challenges within FL. For instance, FedProx [128] introduces weighted aggregation across various clients, though determining appropriate weights for diverse applications remains a complex issue. Alternative strategies involve sharing data between local devices or through server-side proxies [251] to address data heterogeneity, yet this demands an understanding of local data distribution. Strategies such as bounding gradients [240] or incorporating additional noise [105] have been proposed to ensure convergence, albeit at the cost of increased training duration and reduced model precision. GNN models have shown superior results in tasks such as graph or node classification [94, 99, 123, 249], effectively extracting node dependencies within graphs to create graph embeddings. Recent efforts [22, 125, 148, 165, 211] have applied FL to GNN models, such as GraphFL [211] which utilizes a meta-learning-based FL framework for semi-supervised node classification. Yet, the presence of *non-i.i.d.* structural information within graph data poses additional challenges, potentially impairing learning performance [229].

Contribution of this Dissertation

This dissertation proposes a layer-wise dynamic clustering-based FL algorithm, drawing upon certain critical observations. A principal observation is that despite the heterogeneity in data, similar data distribution patterns emerge in smart homes due to the presence of common automation rules among various users. We design a layer-wise clustering-based federated contrastive GNN model for *non-i.i.d.* graph datasets to learn contrastive graph representations. In each cluster, the graph datasets share similar feature information, which exhibits the *i.i.d.* property. In this way, we can reduce the data heterogeneity during the federated learning process in each cluster of clients.

1.1.3: Facial Expression Recognition

Many researchers have proposed to use body sensors that generate physiological signals [32, 175] to recognize emotional states. Yet, such methods require extensive signal analysis over prolonged periods, such as 30 seconds [68], which may hinder efficiency. ExpressEar [208] leverages commercially available ear-worn devices with added inertial sensors to detect facial muscle activities linked to emotions. Similarly, FaceListener [194] transforms headphones into acoustic sensors to measure facial skin changes. However, users need to wear additional devices for emotion detection, which is a notable drawback.

In addition, wireless and mobile sensing technologies have been employed for recognizing behaviors, including daily activities [86, 118, 119, 217, 253], and facial expressions [36]. For example, WiFace [36] discerns facial expressions by analyzing WiFi signal patterns, while Hof *et al.* [91] propose to use mm-Wave radar for facial recognition. These methods, however, necessitate specialized hardware and setup, which may not always be practical.

Acoustic sensing, leveraging ubiquitous speakers and microphones, involves emitting acoustic signals and interpreting the echoes from objects to monitor breathing [233], authenticate users [254], and recognize activities [76, 121, 230]. EchoPrint [254], for instance, uses both acoustic and visual signals for user verification, and TeethPass [230] authenticates users via the sound of teeth clenching captured through earbuds. Similarly, LASense [121] enhances activity sensing precision by refining signal processing techniques. Another application, SonicFace [75], evaluates emotional expressions by analyzing echo patterns, but it cannot obtain fine-grained facial features due to the acoustic signals' frequency resolution constraints.

Contribution of this Dissertation

In this dissertation, we leverage commercial smartphones to emit near-ultrasonic signals (19-23 kHz) directed at the user's face. The smartphone's microphone captures the echo signals reflected off the face, encapsulating details of the facial expressions. Through an analysis of these detailed echo patterns, our developed system, Facer, is capable of identifying six distinct facial emotional expressions: anger, disgust, happiness, fear, sadness, and surprise. We have designed a novel model for acoustic facial expression recognition, which employs contrastive external attention to discern and fortify facial expression features, simultaneously filtering out irrelevant background noise. Furthermore, we introduce a domain adaptation contrastive learning algorithm to reconcile the distribution discrepancies between training and testing datasets, significantly reducing the impact of variability in facial expressions among different users. Our smartphone-based system, Facer, has been rigorously tested in diverse real-world settings, demonstrating its superiority over existing methods by achieving more than a 10% improvement in recognition accuracy, while also offering enhanced mobility and user convenience.

1.1.4: Mobile Activity-Sensing Privacy Protection

Metadata Information Protection

Two principal approaches exist for safeguarding metadata information. The first approach involves adversarial training-based techniques [21, 115, 133, 140, 173], where a discriminator model seeks to deduce sensitive attributes from the features, while a generator model aims to reduce the discriminator's success rate. For instance, DySan [21] obfuscates personal attributes through adversarial training on datasets containing such attributes. The second approach encompasses methods using variational autoencoders and information theory [160, 252], striving to minimize the mutual information between private attributes and their representations, as seen in InfoCensor [252], which addresses both privacy and fairness issues in deep learning.

However, the current body of work exhibits several limitations. Initially, many methods necessitate the disclosure of labeled private data by users for model training, as the optimization loss functions in these models depend on labeled private attributes. Secondly, conventional noise addition techniques compromise not only the metadata but also the utility of activity recognition. Lastly, existing research [21, 133, 140] often overlooks overprivileged issues at the feature level, focusing merely on particular sensitive attributes such as age, gender, or user identity.

Semantic Information Protection

To safeguard semantic details, a simple idea is to generalize the activity labels provided by activity recognition APIs [47, 49]. However, the label generalization alone doesn't ensure privacy protection [79], not to mention that many applications require access to the raw sensor data instead of labels. For example, gaming apps might use motion sensor data to interpret user gestures, and navigation apps might depend on geomagnetic and accelerometer data for directional guidance.

Many mechanisms [20, 30, 79, 143, 171] are designed to determine if activity labels should be disclosed or withheld. Nonetheless, there are two main limitations. First, filter-based methods [30, 79, 171] depend on real-time activity labeling for their decision-making, which is difficult due to human behavior variability and sensor diversity, with limitations noted in platforms such as Android, which only recognizes a limited number of activities [49]. Second, for applications that need raw sensor data, existing privacy mechanisms are too coarse-grained, often removing entire data segments associated with sensitive motions, thereby compromising data utility. For instance, IPShield [30] completely conceals segments of raw data linked to private activities.

Addressing the handling of raw data, multiple obfuscation techniques [60, 89, 198, 218, 248] have been suggested. Anonymization methods [124,139,198] primarily focus on static databases to safeguard tabular data against specific types of attacks. These methods, however, are generally too imprecise for the nuanced issues in time-series data. While differential privacy (DP) [60] introduces noise to data for statistical analyses, it's not ideally suited for interpreting streaming sensor data. Dataset synthesis models based on DP [69,218,246–248], such as PrivSyn [248], are designed for tabular data generation under DP but do not directly address the nuanced overprivileged challenges present in mobile sensing data.

Contribution of this Dissertation

In this dissertation, we propose a multi-grained data generation system, Hippo, for user-controlled information disclosure. Specifically, we introduce a novel system that leverages a noise diffusion process to obscure private attributes, maintaining activity recognition capabilities without necessitating user-provided private data and labels. In addition, we have developed a hierarchical latent feature guidance diffusion model for generating multi-grained data, enabling precise control of information release and addressing feature-level overprivileged concerns. The "granularity" is determined based on the features learned at the i-th layer of a stacked convolutional autoencoder. Hippo offers two main advantages: (i) it operates as a self-supervised framework, processing raw sensing data without depending on labeled private attributes, and (ii) it focuses on generalizing individual activities rather than a sequence of activities, eliminating the need for predefined patterns of private activities. For example, it can transform detailed "playing games" data into a more general "sitting" category. This process filters out the specific "playing games" features, allowing Hippo to provide fine-grained control over the data available to overprivileged applications by eliminating fine-grained and sensitive activity features.

1.2: Research Scope

1.2.1: IoT Device Interactive Threat Detection

As a typical scenario in the IoT, the surge in smart home adoption has significantly elevated the importance and necessity of smart home security. As homes become increasingly interconnected with many smart devices— from thermostats, lighting systems, and security cameras to voice assistants and automated appliances—the potential attack surface for cyber threats expands correspondingly. Smart home security is essential not only for protecting the privacy and personal data of individuals but also for ensuring the physical safety and integrity of the home environment.

From the application layer, we study how to manage the configuration data of smart home automation to mitigate interactive threats spanning various closed-source platforms in real-time, especially when it's challenging to enumerate interactive threat patterns in a cross-platform setting. To tackle this problem, our goal is to devise a data-driven methodology that efficiently identifies patterns of interactive threats. While current data mining approaches to threat detection, such as [59, 109,236,257], typically rely on event logs in sequence, they often fall short in capturing the intricate logic of interactions within smart apps. Thus, we propose to use a graph data structure to represent the rich event semantics and the complex interplay of logic within automation control apps. Then, we design the graph neural network model and transfer learning techniques to learn the interactive threat patterns behind the complex device interaction graphs.

1.2.2: Federated Training for Interactive Threat Detection Model

In this dissertation, we study how can we develop an interactive threat detection model that accommodates the significant heterogeneity in datasets without necessitating the sharing of raw data. Given the limited number of devices in a single household, training a specialized and robust threat detection model is not practical. Federated learning (FL) offers a solution by enabling collaborative model training without disclosing raw data. However, dataset heterogeneity can impair the FL model's performance, often resulting in slow convergence and diminished accuracy, particularly when the data distribution is dynamic or unbalanced across different clients [9,45,199]. To address this, we propose a layer-wise clustering-based federated graph contrastive learning framework that enhances the training of a shared vulnerability detection model, distinguishing between normal and vulnerable graphs across *non-i.i.d.* datasets, thereby enhancing generalization capabilities without compromising user data privacy.

In addition, we study how to filter out drifting threat patterns to mitigate the data heterogeneity. The IoT environment is inherently complex, characterized by various device types and potential threats. Despite utilizing FL to train on diverse interaction graph data from multiple households, there remains the possibility of encountering drifting samples that represent either evolved or entirely new vulnerabilities. Thus, we design the statistic-based drifting sample detection method with federated contrastive graph representations. In this way, we can isolate these drifting samples. Furthermore, to pinpoint potential triggers of vulnerabilities, we devise an effective causal analysis strategy by examining different subgraphs within an interaction graph. Specifically, we apply the Monte Carlo beam search (MCBS) [25], a strategic heuristic algorithm for game tree exploration. Then, we propose to employ optimal SHAP values [136] to assess the risk associated with different subgraphs in the interaction graph. The SHAP value can leverage the potential correlational features among IoT devices. This method allows us to identify and follow the most probable information flow chain that could contribute to the observed interactive threats.

1.2.3: Mobile Sensing and Privacy

Mobile activity sensing leverages the various sensors embedded in smartphones and wearables, such as accelerometers, gyroscopes, magnetometers, microphones, and cameras, to collect data about the movements and behaviors of users. In this dissertation, we propose Facer, a system for recognizing facial expressions utilizing near-ultrasound acoustic sensing technology on smartphones. The smartphones can emit near-ultrasound signals (19-23 kHz) directed at the user's face from the earpiece speaker. The microphone captures the echoes reflected off the face, containing information about the user's facial expressions. By analyzing the detailed echo patterns, Facer can identify six distinct facial emotional expressions. Our experiments show that when a user holds the smartphone at a consistent distance (e.g., 20-50 cm), Facer can accurately recognize six universal facial expressions with an accuracy exceeding 85%.

Meanwhile, mobile sensing data often encompasses more details than what is necessary for the intended functionalities of user applications. As a result, mobile applications tend to have excessive access to sensing information, leading to overprivileged issues [201, 237]. To mitigate the overprivileged issues, we have developed Hippo, a system that leverages a generative AI model to reconstruct mobile activity sensing data at various granularity. The primary goal of Hippo is to obscure sensitive attributes and generalize overly detailed activity data. To obscure sensitive information, Hippo introduces a controlled amount of noise into the sensing data using a diffusion model, a type of generative model. When it comes to generalizing activity features, Hippo is designed to generate new data at different levels of granularity, effectively stripping away sensitive activity features. The proposed method allows users to eliminate unintended sensitive details in activity data, focusing only on the essential semantics required for different applications.

1.2.4: Data-centric AI Algorithms

Computer security research encompasses a broad range of methods to address different types of threats and vulnerabilities in various systems, from traditional computer networks to modern AI-driven systems. For example, rules-based methods use predefined heuristics and patterns to detect violations and security threats, but they cannot discover potential new threats. Formal analysis through model checking and symbolic execution requires significant running time and specialized knowledge to implement. In addition, applying formal methods to large systems can be challenging because the complexity of the analysis tends to grow exponentially with the size of the system. Furthermore, formal methods can hardly capture all aspects of a system's behavior, especially human factors or external environmental variables. This can be a disadvantage in dynamic or rapidly evolving environments where flexibility and the ability to adapt quickly are crucial.

AI and machine learning-based methods are used in security for tasks such as anomaly detection and automated threat detection systems. These methods can adapt to new threats faster than traditional methods. First, AI models can analyze large volumes of data and identify patterns that may indicate malicious activity. This capability allows AI systems to detect sophisticated threats that might elude traditional, rule-based security systems, such as zero-day attacks and advanced persistent threats (APTs). Second, AI can process and analyze data at a speed and scale unattainable to human analysts. This is crucial in security contexts where time is often critical to prevent damage. AI systems can monitor thousands of events across multiple vectors simultaneously, providing a scalable solution that grows with the network or system it protects. Third, through predictive analytics, AI can foresee potential security incidents before they occur by identifying the likelihood of threats based on historical data and emerging trends. This proactive approach helps organizations to preemptively tighten their defenses and allocate resources more effectively to high-risk areas. Fourth, AI systems can continuously learn from new data. As they encounter new types of attacks or variations of existing threats, they can adapt and refine their detection and response strategies without human intervention. This self-improving capability is crucial in keeping up with the rapid development of new threats.

Data-centric AI is a methodology that focuses on improving the quality, consistency, and relevance of the data used in training AI models. Considering the performance of AI models is dependent on the training dataset, instead of primarily focusing on modifying the algorithms, we need to pay more attention to the data preparation process, which serves as a key driver of the AI revolution. There are several key aspects of data-centric AI. First, the data quality is crucial for model training. AI models learn to make predictions based on the data they are trained on. If this data contains noise, biases, or corruption, the model will inherently learn these flaws, leading to inaccurate and unreliable outputs. Second, data quantity is important to enhance the generalization and robustness of AI models. Enhancing the diversity and volume of training data through techniques such as synthetic data generation or transformations can help improve the robustness of AI models. Third, data collection strategies are essential for designing data-centric AI algorithms. We need to collect more representative or comprehensive data sets to cover a wider variety of scenarios that the model might encounter in real-world applications. Meanwhile, we need to check for and mitigate biases, which can otherwise lead to unfair, unethical, or harmful decisions. Incorporating continuous feedback into the training process can refine and update the data sets and model accuracy over time.

1.3: Organization

The following sections of the dissertation are structured as follows. Chapter 2 delves into the proposed interactive threat detection model, which includes IoT data fusion, interactive threat pattern learning, and threat knowledge transfer learning. Chapter 3 introduces the designed layer-wise clustering-based federated learning approach for training the threat detection model, and details the proposed Monte Carlo beam search-based method for threat root cause analysis. Chapter 4 presents the proposed contrastive attention-based acoustic sensing model for six facial expression recognition, which reveals a new potential privacy leakage channel using smartphones. Chapter 5 introduces the designed multi-grained mobile activity sensing data generation model to address both metadata-level and feature-level mobile sensing data overprivileged issues. Finally, Chapter 6 concludes this dissertation and outlines opportunities for future research.

CHAPTER 2: GRAPH LEARNING FOR INTERACTIVE THREAT DETECTION[1]

2.1: Introduction

Smart homes are increasingly equipped with intelligent devices such as smart locks, smart plugs, and smart ovens, enhancing the living experience through automation. Homeowners can integrate various systems to leverage cross-platform automation functionalities, considering aspects such as device compatibility, centralized voice control, and user-friendly operation. A recent study [38] indicates that 82.4% of smart home setups utilize multiple automation rules for a single device, and 62.4% of users deploy more than one platform in their residences. The integration of different platforms is facilitated by universal application programming interfaces (APIs), enabling apps from distinct platforms to interact seamlessly. For instance, IFTTT [97] can link with other ecosystems such as SmartThings [189], Amazon Alexa [5], and Home Assistant [10].

Nonetheless, the interconnections between users, devices and the environment can lead to undesirable or insecure issues, referred to as interactive threats. These threats can arise from various sources, including user misconfigurations [93, 205] or external attacks [1, 245], particularly when multiple systems are configured simultaneously. For example, a conflict might occur in a smart home if a Home Assistant app sets a rule to *"open the window if smoke is detected"* which conflicts with another rule to *"close the window when the outside temperature is high"*, potentially hindering the window's ability to open in response to smoke. This highlights the critical need to manage the interaction logic among automation rules to avoid potential hazards and property damage.

Various techniques have been devised to analyze app interactions, but when confronted with

[1]This chapter is based on previously published work by Guangjing Wang, Nikolay Ivanov, Bocheng Chen, Qi Wang, ThanhVu Nguyen, and Qiben Yan titled "Graph Learning for Interactive Threat Detection in Heterogeneous Smart Home Rule Data" published in the Proceedings of the ACM on Management of Data. DOI: 10.1145/3588956 [215].

closed-source, heterogeneous platforms, two main challenges arise. The first major challenge is that many prevalent platforms, such as IFTTT and Alexa, are closed-source. This fact makes methods dependent on source code analysis or code instrumentation futile [3, 55, 203]. Consequently, these proprietary environments hinder the effectiveness of existing heuristic strategies that are reliant on analyzing or instrumenting source code [2, 27, 28, 39, 54, 55, 70, 154, 203].

The second challenge is the complexity of interactions across various platforms. Many approaches that are based on security policies [28, 70, 222] operate under the assumption that all automation rules are executed within a single platform [38]. These methods are limited to intra-app and inter-app analyses within a single platform environment, such as the open-source platform SmartThings [189]. In addition, existing methods require predefined threat patterns and security policies. Such approaches are often inadequate to encompass the diverse array of threat types stemming from intricate rule interactions across different platforms.

In this dissertation, we tackle an important but unexplored research question: how can we manage the configuration data of smart home automation systems to avoid interactive threats spanning numerous closed-source platforms? In a cross-platform setting, it is challenging to enumerate all possible interactive threat patterns. To resolve this question, we propose a novel data-driven method aimed at efficiently identifying interactive threat patterns. While current data mining strategies for detecting threats typically rely on event logs [59, 109, 236, 257], these methods do not fully capitalize on the intricate interaction logic inherent in smart apps. This is primarily due to the difficulty of representing the semantics of events and the sophisticated interaction logic through the sequence structure of time-series data.

Graph neural network (GNN) models [71, 96, 108, 228] have experienced widespread success in various tasks such as a graph or node classification problem, link prediction, and graph matching. GNNs are adept at capturing both the explicit features of nodes and the implicit structural features of graphs. In the context of smart home automation, the interactions among different platform rules can be structured into a graph, where each node represents a rule, and the edges depict the interactions between these rules. Node features can be enriched using semantic-aware embeddings derived from natural language processing (NLP) techniques [18]. We model interactive threat analysis in smart homes as a graph representation learning task using GNNs and introduce **Glint**, a framework called **G**raph **l**earning for **int**eractive threat analysis, aimed at safeguarding heterogeneous smart home environments.

There are three main challenges in designing Glint for interactive threat detection:

First, there is a notable absence of graph data with ground truth labels for identifying interaction patterns on closed-source smart home platforms. Nevertheless, smart home platforms have comprehensive app descriptions, which include device control rule examples. In addition, the event logs are available in most smart home apps, which reflect timestamps and device states. By integrating rule descriptions and event log data, we can create a dynamic interaction graph. Furthermore, insights from previous research on open-source platforms [55, 70, 203] provide valuable expert knowledge on vulnerable device interactions, enabling us to construct and label large interaction graph datasets. Thus, we can model interactive threat analysis as a supervised learning problem. This approach is beneficial for its accuracy, leveraging labeled data for learning, and efficiency during the testing phase with a trained model.

Second, interaction patterns may evolve, posing the issue of drifting samples due to new or changing threat types, making it challenging to catalog all possible interactive threats. To address the problem, we introduce the ITGNN model, which incorporates a contrastive learning loss to discern graph embeddings, allowing for the identification of drifting samples. This data-driven strategy, utilizing a GNN model to uncover potential threat patterns from graph datasets, surpasses the limitations of static analysis and dynamic testing, which depend on predefined patterns, thus enhancing security policy coverage with Glint.

Third, data availability can significantly vary across platforms, potentially impacting graph representation learning outcomes. Despite this variability in data formats, the underlying semantics of rules across different platforms are similar since they all aim to control smart devices. To overcome data scarcity on certain platforms, we propose a cross-domain graph transfer learning approach, enabling the transfer of knowledge across heterogeneous platforms and boosting model efficacy on data-limited platforms.

We conducted a comprehensive evaluation of Glint using datasets crawled from five platforms, encompassing over 48,000 interaction graphs. Our results demonstrate that Glint is proficient in identifying interactive threats both within individual platforms and across multiple platforms. Notably, within the SmartThings platform, Glint achieved a detection accuracy of 100%. For detecting threats in diverse interaction graphs, it attained an accuracy of 95.5% and an F1 score of 95.6%. Furthermore, through the analysis of drifting samples, Glint successfully identified four new interactive threat types in custom blueprints on the Home Assistant platform, which we have named “action block”, “action ablation”, “trigger intake”, and “condition duplicate”.

In summary, we make the following contributions:

- We have created the first large interaction graph datasets with verified ground truth labels by examining the trigger-action relationships among rules from closed-source platforms.
- The ITGNN model is developed to facilitate the learning of interaction patterns across diverse platforms. This model utilizes contrastive learning to identify drifting samples and applies transfer learning to improve the generalization capabilities of GNN models.
- We have extensively evaluated Glint in a real-world environment. The evaluation results indicate that Glint surpasses existing state-of-the-art methods in detecting interactive threats across platforms, improving precision by 12.8% and recall rates by 12.6%.

2.2: Problem Definition

We initiate our discussion by presenting a running example of interaction graphs, which sets the stage for a deeper understanding. Then, we provide a formal definition of interactive threats. The process of identifying interactive threat patterns is then elaborated upon, highlighting how graph representation learning plays a crucial role in the threat discovery process.

Table 2.1: Specific rule contents of nodes, where the index is the node ID in Figure 2.1.

Index	Platform	Automation Rule Description
1	SmartThings	Turn off lights if playing movies.
2	SmartThings	If the outdoor temperature is between 65 °F and 80 °F, open windows after sun rise.
3	SmartThings	If outdoor temperature is below 60 °F, then close windows.
4	SmartThings	Turn on the air conditioner when temperature is above 85 °F.
5	IFTTT	If air conditioner is on, then close windows.
6	IFTTT	If the smoke alarm is beeping, then open the window and unlock the door.
7	IFTTT	If motion is detected, turn on lights.
8	IFTTT	If motion is detected, open the door.
9	Amazon Alexa	Lock the door if all lights are turned off.

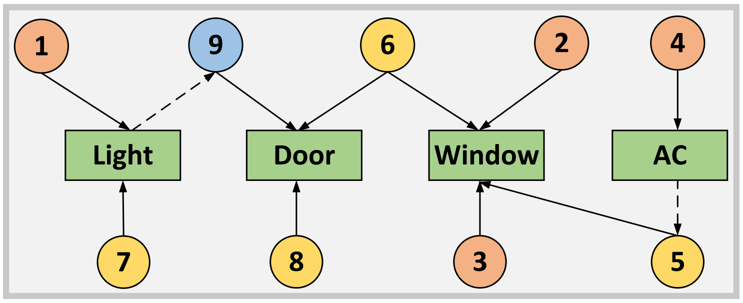


Figure 2.1: An interaction graph, where nodes (in circle shape) are rules from three platforms.

2.2.1: Interaction Graph

An interaction graph serves as a conceptual model that illustrates the relationships among various rules from different platforms. For instance, as depicted in Figure 2.1, an interaction graph encompasses rules from SmartThings, IFTTT, and Alexa. The specifics of each rule within the graph are detailed in Table 2.1. These automation rules encapsulate a "trigger-action" logic, indicating that the activation of one rule prompts the execution of another. For example, the command "*Alexa, play movies*" is linked with Rule 1 through a "trigger-action" relationship.

Rules can be interconnected through various devices and physical channels, such as temperature or humidity. Additionally, an "action-trigger" relationship can exist between rules, where the action of one rule initiates the execution of another. For instance, the interaction between Rule 1 and Rule 9 is mediated through the action of turning off lights, which subsequently triggers the locking of a door when movies are played. Interaction graphs encapsulate interactions between devices, users, and the environment based on “trigger-action” logic. The “trigger-action” and “action-trigger” connections exemplify the causal relationships within the configuration data of automation rules.

Suppose there are n rules that are deployed, then the number of distinct interaction graphs is fixed. For example, if there are ten smart home apps each with a single rule, only one interaction graph will exist in real time, defined by the “action-trigger” relations among the rules. The absence of a direct relationship between any two rules means an edge will not connect their corresponding nodes in the graph. Thus, while the total number of rules impacts the complexity and size of an interaction graph, it does not directly alter the count of interaction graphs in real-time.

2.2.2: Threat Model

Interactive threats in smart home environments are essentially the abnormal interactions among various devices, users, and their surroundings. These interactions, harboring threats or anomalies, can lead to unwanted behaviors or significant security and privacy concerns. We mainly consider the threat scenarios where smart home automation control rules interact with each other to jeopardize the home automation system. We define three general scenarios where an adversary can implement attacks to exploit interactive threats:

- Targeted compromise: An attacker can attack a device, such as a voice assistant to control smart devices by executing inaudible voice commands [1, 234, 245]. An attacker can also trick the user into installing malicious apps to trigger unexpected interactive actions.
- Interaction abuse: An attacker can use existing exploitable interaction chains to subvert the intent of original home automation [66, 159]. To achieve that, the attackers can try to trigger an automation rule, which leads to unexpected interactions among devices.
- Misconfiguration: When deploying trigger-action rules, the users may misconfigure their devices, introducing vulnerabilities in the home automation [93, 205]. For instance, the elderly and children can accidentally trigger some vulnerable events via voice assistants, leading to undesirable interactions.

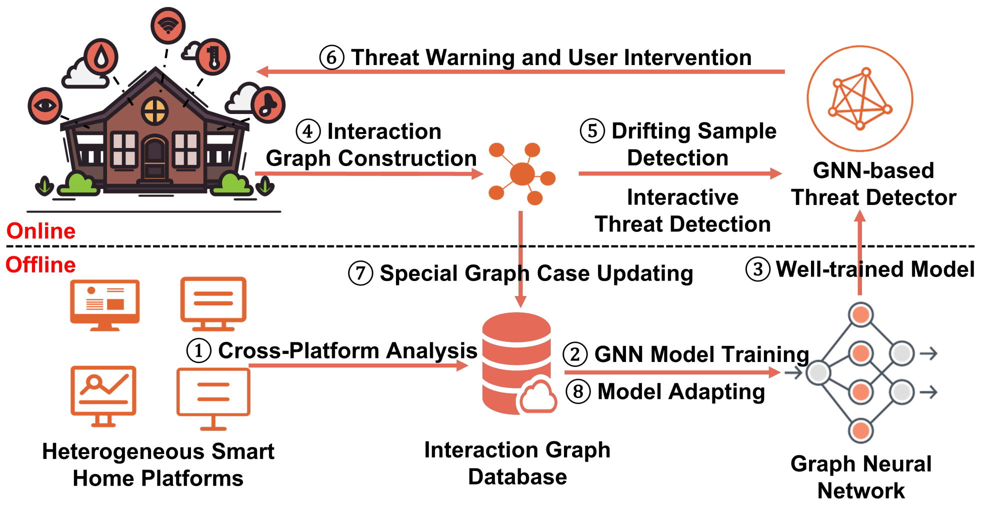
All the above attack scenarios cause interactive threats. An adversary exploits flaws in the interactions among various components to achieve attack goals, such as entering a house without permission, causing property damage by fire, or stealing private information. Note that we assume the smart home IoT platforms and app descriptions are trustworthy. Glint aims to detect interactive vulnerabilities caused by the above attacks, fortifying IoT security in smart homes.

2.2.3: Problem Formulation

We formally define the task of managing automation data to avoid interactive threats as a graph representation learning problem. Given an interaction graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}\}$, the nodes $\mathcal{V}$ are rules from different platforms, the edges $\mathcal{E}$ are correlations among different rules such as "trigger-action" or "action-trigger". Each node $v \in \mathcal{V}$ has a feature vector $x_v \in \mathcal{X}$, which is a word-embedding or sentence-embedding that encodes semantic information. For example, the word "sunset" is encoded as a 300-dimension embedding $[0.60014, \ldots, 0.35422]$ with an NLP library spaCy [195]. We derive the averaged word embedding of each word as the rule-level embedding, which is a node feature. While node features in homogeneous graphs come from the same feature space to better characterize rules from homogeneous platforms, node features in heterogeneous graphs come from distinct feature spaces. Given an interaction graph dataset $\{G\}$, the goal is to design a model $f_w : x^{\{G\}} \rightarrow \mathbb{R}^d$ to learn a d-dimensional representation of the graph G. The model f_w can be different learning tasks, such as drifting sample detection and binary classification. The learned graph representation preserves the structural and semantic information [96], which is associated with the interactive threat patterns.

2.3: System Design

In this section, we delve into the design and application of Glint, outlining its back-end (offline) and front-end (online) components. We elaborate on the four pivotal elements of Glint: (i) The process of constructing the interaction graph dataset is detailed, explaining how we gather, structure, and label the data to create a comprehensive representation of smart home interactions. (ii) The



⑥ Threat Warning and User Intervention
④ Interaction Graph Construction
⑤ Drifting Sample Detection
Interactive Threat Detection
GNN-based Threat Detector
Online
Offline
⑦ Special Graph Case Updating
③ Well-trained Model
① Cross-Platform Analysis
② GNN Model Training
⑧ Model Adapting
Heterogeneous Smart Home Platforms
Interaction Graph Database
Graph Neural Network

Figure 2.2: Glint builds an interaction graph for a house, and uses a well-trained GNN model to automatically discover interactive threats and generate threat warnings.

ITGNN model is introduced as our chosen method for graph representation learning, detailing its architecture and how it processes the interaction graph to learn meaningful representations. (iii) We describe the algorithm developed to detect drifting interaction patterns, highlighting its role in identifying evolving or new threats within the smart home environment. (iv) We discuss our strategy for augmenting the ITGNN model's generalization capabilities, which involves transferring learned knowledge across different, heterogeneous smart home platforms to improve model robustness and adaptability.

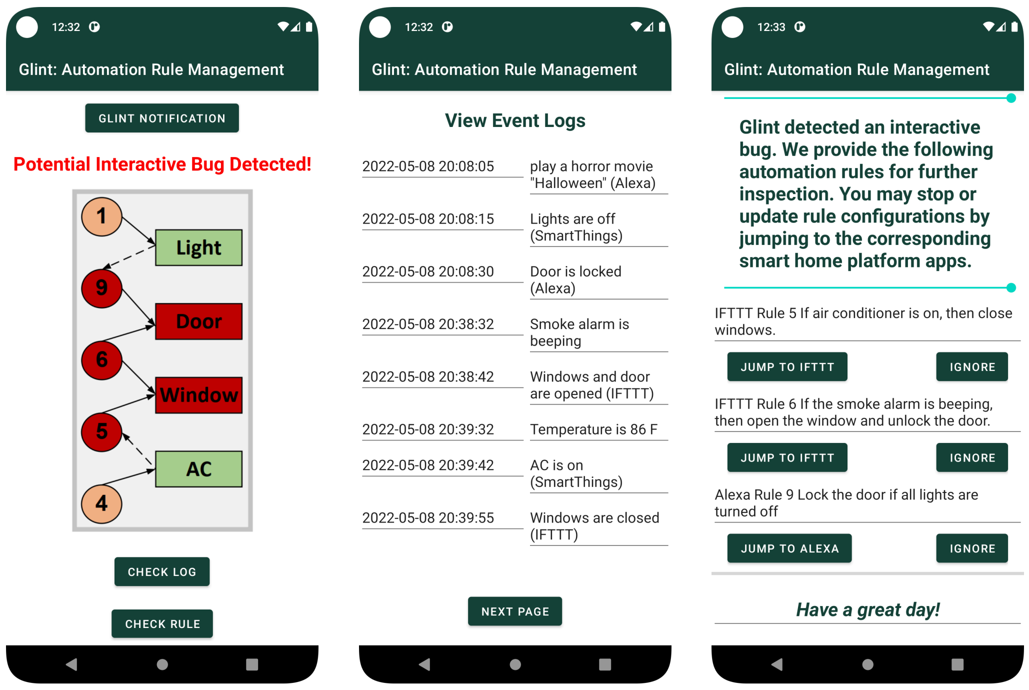
2.3.1: System Overview

Glint operates through two primary phases: offline model training and real-time interactive threat detection. The workflow of smart home automation data analysis is depicted in Figure 2.2. In the offline stage: ①: Glint conducts an analysis of potential intra-platform and inter-platform interactions to construct interaction graphs, which are then aggregated into a comprehensive interaction graph dataset and stored in a database. ②: The GNN model, tailored for this task, is trained on the interaction graph dataset, utilizing contrastive learning and a classification approach. ③: Transfer learning is applied to the trained model to enhance its generalization capabilities, culminating in the development of a GNN-based interactive threat detection tool.

In the online stage: ④: Utilizing the trigger-action logic derived from rules and device statuses from event logs, Glint can dynamically generate interaction graphs in real time. ⑤: The freshly constructed graph undergoes a two-step analysis: initially, it's processed by the contrastive learning-based drifting sample detector, followed by scrutiny from the interactive threat detector (classification model). ⑥: Should an interactive threat be identified, Glint issues an alert and prompts for user intervention. ⑦ & ⑧: In cases where drifting samples are detected or when users identify non-threatening graphs as false alarms, these specific graph cases are noted, and the model is fine-tuned to better align with user expectations and improve threat detection accuracy.

In the offline stage, a generic GNN model is developed utilizing graph datasets from diverse platforms. This model, created by a security solution provider, can be accessed via cloud services. Each user utilizes the same offline component, ensuring consistency in the model's foundational training. For practical application, Glint is integrated into the smart home hub, simplifying the process for users who then primarily interact with the online stage. This feature enables Glint to play a crucial role in scanning for interactive threats during the initial configuration of a smart home system. As the system operates and gathers deployment setups and event log data, the GNN model undergoes fine-tuning to align with individual user preferences. Additionally, a companion application, Glint, can be installed on smartphones. This app is designed to alert users to potential interactive threats detected by Glint, facilitating user engagement and intervention when necessary. This interactive process ensures that the smart home system remains secure and tailored to the user's specific requirements and preferences.

Imagine a user sets up a series of rules detailed in Table 2.1. For the sake of simplicity, let's assume that Rules 2, 3, 7, and 8 are inactive during a certain period. A real-time interaction graph is formulated by examining the rule descriptions that have been deployed and the runtime event logs. When Glint identifies an interactive threat, it alerts the user, as depicted in Figure 2.3a. In this interaction graph, the potential sources of the threat (e.g., specific nodes and associated devices) are highlighted in red to help reduce the amount of information the user has to process. These potential sources can be pinpointed using GNN explanation tools like PGExplainer [137],


12:32

Glint: Automation Rule Management

GLINT NOTIFICATION

Potential Interactive Bug Detected!

CHECK LOG

CHECK RULE

12:32

Glint: Automation Rule Management

View Event Logs

2022-05-08 20:08:05	play a horror movie "Halloween" (Alexa)
2022-05-08 20:08:15	Lights are off (SmartThings)
2022-05-08 20:08:30	Door is locked (Alexa)
2022-05-08 20:38:32	Smoke alarm is beeping
2022-05-08 20:38:42	Windows and door are opened (IFTTT)
2022-05-08 20:39:32	Temperature is 86 F
2022-05-08 20:39:42	AC is on (SmartThings)
2022-05-08 20:39:55	Windows are closed (IFTTT)

NEXT PAGE

12:33

Glint: Automation Rule Management

Glint detected an interactive bug. We provide the following automation rules for further inspection. You may stop or update rule configurations by jumping to the corresponding smart home platform apps.

IFTTT Rule 5 If air conditioner is on, then close windows.

JUMP TO IFTTT IGNORE

IFTTT Rule 6 If the smoke alarm is beeping, then open the window and unlock the door.

JUMP TO IFTTT IGNORE

Alexa Rule 9 Lock the door if all lights are turned off

JUMP TO ALEXA IGNORE

Have a great day!

(a) Notification. (b) Event logs. (c) Operation.

Figure 2.3: The usability demonstration of Glint.

SubgraphX [241], or FexIoT [214], which determine the crucial subgraphs or nodes influencing the GNN's prediction. By considering the user's daily appliance usage habits and analyzing the event logs shown in Figure 2.3b alongside the potentially vulnerable rules illustrated in Figure 2.3c, the user can discern if the rule interaction is undesirable. If it is, they have the option to halt or modify the rule configurations by adjusting the relevant smart home platform apps, ensuring the system's alignment with their preferences and enhancing security.

2.3.2: Building Interaction Graph Dataset

We first build interaction graph datasets using rule descriptions in the offline stage. Then, we construct real-time interaction graphs with deployed rules and event logs in the online stage. We consider exploring IoT interactions in closed-source and cross-platform environments. The design of the NLP pipeline facilitates information access in the closed-source platforms.

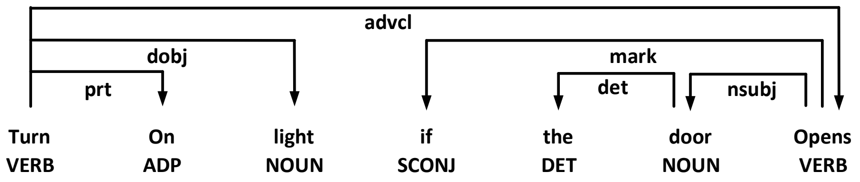


Figure 2.4: An example of dependency parsing.

Algorithm 1: Home Automation Rule Feature Extraction

```
    Input: IoT Automation Rule Trigger Set $T_S$, Action Set $A_S$
    Output: Rule Correlation Feature Vector $V$
1   foreach $T \in T_S$ and $A \in A_S$ do
2       $[nouns, verbs]_T = PoS(T)$
3       $[nouns, verbs]_A = PoS(A)$
4       $V1 = DTWDistance([nouns, verbs]_T, [nouns, verbs]_A)$
5       $V2 = binaryRelation([verbs]_T, [verbs]_A)$
6       $V3 = binaryRelation([nouns]_T, [nouns]_A)$
7       $V4 = \overline{E}_T + \overline{E}_A$    ▷ $\overline{E}_T$ and $\overline{E}_A$ are averaged word embeddings of a trigger and an
          action, respectively.
8   end
9   return $[V1, V2, V3, V4]$
```

Rule Correlation Discovery

The primary obstacle in constructing interaction graph datasets lies in identifying the correlations among various rules. The plethora of noisy, inconsistently formatted data complicates the task of establishing rule correlations across diverse platforms. Drawing inspiration from existing research [197, 200, 222], we employ natural language processing (NLP) methods to extract meaningful semantic features from the publicly accessible descriptions of apps. Given the extensive potential “action-trigger” combinations, we approach correlation identification as a binary classification task, aiming to minimize the need for extensive manual labeling.

We design the smart home automation rule feature extraction algorithm as shown in Algorithm 1. The input is a set of trigger phrases and action phrases, and the output is correlation feature vectors. We implement part-of-speech (POS) tagging and syntactic element extraction (lines 2-3). POS tagging follows context and word definition to recognize the main task, trigger object, and action object. For example, in Figure 2.4, the dependency parser recognizes the root verb *Turn* as the main task. The dependency edge connects other syntactic words to the root verb, which indicates a grammatical relation between the trigger object and the action object. We focus on direct objects (dobj), nominal subjects (nsubj), compounds and modifiers, nominal and passive nominal subjects, and clausal complements, based on which we extract the main task, objects, and their properties in one sentence. Moreover, we discard the named entity since a named entity may modify two different objects, which will reduce the uniqueness of features. For example, *Wyze Cam* and *Wyze Thermostat* are two different objects, but *Wyze* will add bias when computing the semantic similarity.

We compute the numerical context features based on the linguistic elements. We calculate the verb similarity and object similarity in the trigger-action pairs (line 4 in Algorithm 1). We apply dynamic time warping [169] to compute similarity because the number of verbs or objects in trigger and action sentences varies. Then, we compute the binary semantic features (lines 5-6). We analyze whether the verb or object words in trigger-action pairs have synonym or hypernym relations and whether the object words in trigger-action pairs have meronym (i.e., a constituent part of an object) or holonym (i.e., a part of meronym name) relations. These relations can better reflect the generic semantic relationships between triggers and actions. Finally, we compute the trigger-action pair embedding (line 7) by summing the averaged word embeddings in corresponding triggers and actions. These are unique features of "trigger-action" rule pairs, which allow classifiers to learn the internal characteristics of interactions. We compose the above semantic features into feature vectors to train a model that detects correlations among "action-trigger" correlation pairs. In this way, we can build correlation pairs among different rules, which are formulated as a binary classification problem. If the given action can lead to the invocation of the trigger, it is labeled "true". Otherwise, it is labeled "false".

Offline and Online Graph Construction

In the offline stage, there are two phases to building interaction graphs. The input dataset is a set of crawled rule sentences, which provide rich trigger-action interaction logic. First, based on the rule correlation discovery model, we predict the remaining unlabeled rule pairs. In this way, we obtain "action-trigger" correlations between two rules. In the second phase, based on the correlation results, we randomly select and concatenate the "trigger-action" and "action-trigger" rule interactions to form interaction graphs, which makes generated graphs less prone to bias. We use the DGL [221] library to build and store the graphs, which are accessible with DGL APIs.

In the online stage, we consider the scenarios when one house is equipped with single or multiple platforms. With our designed offline rule correlation discovery classifier, we can first build a complete interaction graph from all rules deployed in a user's smart home, as shown in Table 2.1. However, it cannot reflect the actual interactions among smart devices in real-time because such graphs lack the triggers' chronological ordering information. Fortunately, event logs contain three basic elements: time, object, and the current status of the object, which can reflect real-time device status as shown in Figure 2.3b. The event time helps determine the event sequences. Users can set up the device name and location when they deploy and configure devices. Thus, we obtain semantic information such as device types, locations, and device states to differentiate among different devices. Then, we match the device type and current status with the graph constructed from the deployed rules. In this way, we can remove the unrealistic "trigger-action" pairs such as the second rule (action) happening before the first rule (trigger).

The temporal dimension is not only integrated into the NLP embeddings but is also used for pruning graphs. First, we encode the temporal semantics (e.g., "sunset, at midnight") into embeddings, such that the ML models could learn the temporal features. Second, in the online stage, we use the triggers' chronological ordering information in event logs. For example, we can set a time interval (e.g., 3 hours) and use the timestamp to remove any unrealistic pairs. In this way, although two rules (nodes) could potentially interact, there is no edge between them in a real-time interaction graph because of the disjoined occurrence time. The temporal information allows the model to better learn "action-trigger" correlations. Besides, with temporal information, we can remove unrealistic correlations and reduce the interaction graph edges, making it convenient for users to inspect device interactions. We acknowledge that there could be opportunities for enhancement from both data and model perspectives when considering the temporal dimension.

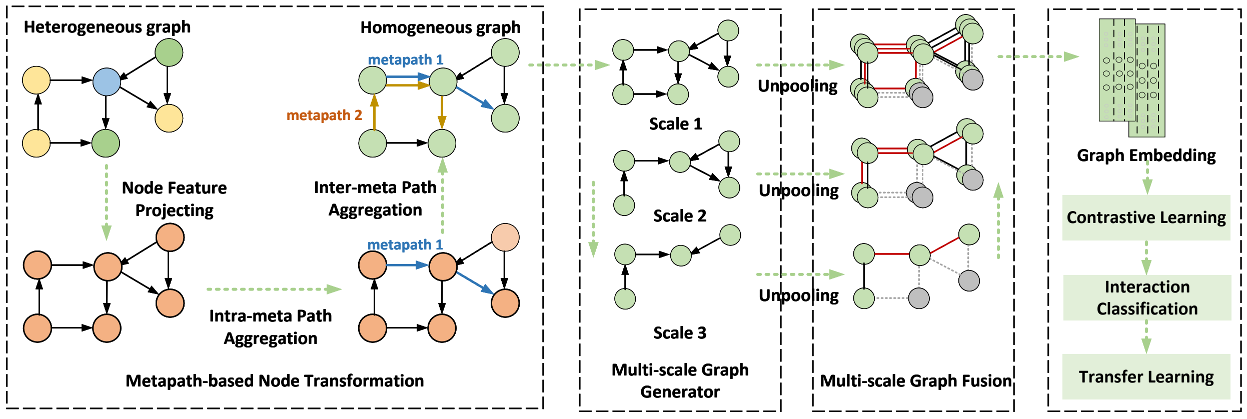


Figure 2.5: The proposed ITGNN model for interaction graph representation learning.

2.3.3: Graph Representation Learning

In this section, we model the interactive threat analysis as a graph learning problem to automatically learn interaction patterns. Considering the different types of information linked to nodes, a single type of feature representation cannot represent the whole graph due to the differences in types and dimensionality. Existing GNN models [71, 96, 108, 228] either target homogeneous graphs or heterogeneous graphs with different designs. Besides, the interaction patterns could exist at different scales in a heterogeneous interaction graph, e.g., at the different subgraph levels. In this chapter, we design **ITGNN**, a unified model for **I**nteractive **T**hreat analysis based on **GNN**, which learns interactive threat patterns from different scales of an interaction graph.

Graph Representation Model Design

We design the interaction graph representation learning process as shown in Figure 2.5, and the ITGNN model is detailed in Algorithm 2. Other input parameters in Algorithm 2 include node type set $\mathcal{A}$; the number of node V_A; node weight matrix W_A; node feature vector $x_v \in X$; metapath set $\mathcal{P}$; the number of layers L; the number of scales D; weight matrix for linear transformation of each metapath-specific node vectors M_A and b_A; activation function σ; aggregation function Φ. ITGNN takes as input a graph $G(V, E, X)$ and corresponding parameters, and outputs a d-dimensional graph representation z_g. ITGNN considers interaction features from both neighborhoods of a vertex and multiscale structures of a graph.

```
Algorithm 2: ITGNN Graph Representation Learning
Input: interaction graph $G$ and other input parameters
Output: graph representation $z_g$
1  if $V_T > 1$ then
2      for $A \in \mathcal{A}$ do
3          $h_v = W_A \cdot X_v^A$
4          for $P \in \mathcal{P}_{\mathcal{A}}$ do
5              for $v \in V_A$ do
6                  $h_v^p = \frac{1}{|N_v^p|} \sum_{v \in N_v^p} (\frac{1}{|v|} \sum_{|v|} h_v), \forall v \in p(v, u_1)$
7              end
8          end
9          $s_{p_i} = \frac{1}{|V_A|} \sum_{v \in V_A} sigmoid(M_A \cdot h_v^{p_i} + b_A)$
10         $\beta_{p_i} = Softmax(q_A \cdot s_{p_i})$
11         $h_v^{p_A} = \sum_{p \in p_A} \beta_p \cdot h_v^p$
12     end
13     $G_m = G(V, E, \{h_v^{p_A}\})$
14 end
15 for $l = 1 \ldots L$ do
16     for $d = 1 \ldots D$ do
17         $h_v^d = VIPool(G_m)$
18         $h_v = h_v || h_v^d$
19     end
20     $h_v^{(L_i)} = \sigma(h_v^{(L_i - 1)}, \Phi(h_u^{(L_i - 1)}; u \in N_v))$
21 end
22 return $z_g = readout(h_v; v \in V)$
```

First, we project all nodes' features in heterogeneous graphs to the same feature space and then aggregate intra-metapath and inter-metapath information, called metapath-based node transformation. A metapath describes a composite relation between a series of node types, and the metapath instance is a sequence of nodes in a graph following the metapath schema [71]. Unlike homogeneous graph classification, we need to consider how to aggregate various types of nodes into the readout function. Inspired by MAGNN model [71], we first project heterogeneous node features to the same embedding space (line 3). For a node v of type A, we have the projected node embedding as h_v. Then, for each target node, we average the node features of metapath instances into a single vector (lines 5-7). In line 6, N_v^p is metapath-based neighborhoods of node v, $p(v, u_1)$ is a metapath instance, and v is the mapping target.

Second, we use the attention mechanism to aggregate the inter-metapath information (lines 9-11), where s_{p_i} is the summarized metapath of $p_i \in p_A$, q_A is the attention vector for node type A and β_{p_i} is the relative importance of each metapath p_i for the targeting node v. The interactive threats are usually caused by a subset of graph metapath instances. Different instances will contribute to the final graph embedding to different extents. In this way, we transform the node embedding features while retaining the graph structure to form homogeneous-type graphs (line 13).

Third, we extract comprehensive features from multiple scales of a graph by incorporating the vertex infomax pooling (VIPool) module [123] into ITGNN, which is called a multi-scale graph generator and fusion. We input the homogeneous-type graph and apply the VIPool [123] to generate multi-scale graphs (lines 15-21), from which we extract the features. The multi-scale graphs are propagated via the TAG [58] convolution layer, which does not need to approximate graph convolution. By vertex pooling, we select and keep vertices that contain high mutual information with neighborhoods, which can well express local subgraphs. We concatenate different scales of features as $h_v = h_v^0 || \cdots || h_v^{d-1} || h_v^d$ (line 18). h_v^d is the d_{th} scale of the graph features, $||$ represents concatenation. Fully connected layers are connected to fuse multi-scale features.

The ITGNN graph representation learning can be used for different tasks. If the interaction graph is heterogeneous, it will first map node features from different feature spaces into the same space. Then, Glint can map the homogeneous graph into graph embedding. Next, we feed the generated graph embedding to the drifting sample detection module and classification module to identify the presence of an interactive threat.

Drifting Interaction Pattern Detection

The concept drift is an important problem when modeling heterogeneous data from different IoT platforms. Given that the interactive threat labeling is based on a set of known interactive threat patterns, it is conceivable that the learned model could have false negatives. This is because the testing interaction graph distribution may drift away from that of the training dataset due to the new smart device automation rules and the presence of various attacks. As a result, there will be a shift in the decision boundary [73] of the supervised or semi-supervised machine learning models,

```
Algorithm 3: Drifting Interaction Pattern Detection
Input: Training interaction graphs $g_i^{(j)}$, $i = \{0, 1\}$ is binary classes, $j = \{1, \cdots, n_i\}$, $n_i$ is
       the sample number in class $i$, testing data $x^{(m)}$, $m = \{1, \cdots, M\}$, $M$ is the total
       number of testing samples, ITGNN-C model $f$
Output: Drifting degree for each testing sample $A^{(k)}$
1  for $i \in \{0, 1\}$ do
2      for $j = 1$ to $n_i$ do
3          $z_i^{(j)} = f(g_i^{(j)})$                    ▷ The latent representation of $g_i^{(j)}$
4      end
5      for $j = 1$ to $n_i$ do
6          $d_i^{(j)} = ||z_i^{(j)} - \frac{1}{n_i} \sum_{j=1}^{n_i} z_i^{(j)}||_2$
7      end
8      $MAD_i = median(|d_i^{(j)} - median(d_i^{(j)})|)$
9  end
10 for $m = 1$ to $M$ do
11     for $i \in \{0, 1\}$ do
12         $d_i^{(m)} = ||f(x^{(m)}) - \frac{1}{n_i} \sum_{j=1}^{n_i} z_i^{(j)}||_2$
13         $A_i^{(m)} = \frac{|d_i^m - median(d_i^{(j)})|}{MAD_i}$
14     end
15     return $min(A_i^{(k)}), i = 1, \cdots, N$
16 end
```

which will degrade the model performance. Since the classification models are mostly based on the i.i.d. assumption, the detection and filtering of drifting samples could help reduce the classification model's false positives and negatives.

Therefore, we need to detect the interaction graph samples that drift from existing classes. Here, we design ITGNN architecture with contrastive learning loss, called ITGNN-C, to learn a distance function to discover drifting samples. Taking advantage of our labeled graph dataset, contrastive learning can achieve better performance compared to unsupervised methods [235]. The basic idea of contrastive learning is to enlarge the distance among samples with different labels and reduce the distance among samples with the same label. Given a set of samples x_i and the corresponding label y_i, the contrastive loss takes a pair of samples (x_i, x_j) as input and tries to learn a function f_θ.

The formula is written as follows:

$$\mathcal{L}_{\mathrm{c}}(\mathbf{x}_i, \mathbf{x}_j, \theta) = \mathcal{T}[y_i = y_j] \| f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j) \|_2^2 + \mathcal{T}[y_i \neq y_j] \max(0, \epsilon - \| f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j) \|_2)^2, \quad (2.1)$$

where $\mathcal{T}[y_i = y_j]$ means the value is 1 if two graph samples have the same label, otherwise is 0. $\mathcal{T}[y_i \neq y_j]$ means the value is 1 if two graph samples have different labels, otherwise is 0. ϵ is the upper bound distance to avoid the exceptional contribution of some dissimilar pairs. The ITGNN-C model augmented with contrastive loss can map samples from each class to a compact region in the latent space. After training the ITGNN-C model with labeled data, we can apply it to discover drifting samples. Given a list of testing samples x_t, with respect to the current classes in the training data, inspired by CADE [235], we evaluate if x_t is a drifting sample as shown in Algorithm 3.

We have binary classes "normal" and "threat", denoted as 0 and 1, respectively. We first generate the latent representations of training graph samples and compute the mean values of the representations as the centroid of each class (lines 2-4). We calculate the median of the absolute deviation MAD_i within each class i (lines 5-9). For testing, we generate representations of the testing samples and compute each test's drifting degree (lines 10-16). We recognize a potential drifting sample by comparing A^k with a threshold T_{MAD}, which is set as 3 empirically [114]. Users or security analysts can further investigate the drifting samples and conduct retraining.

Vulnerable Interaction Classification

Besides designing contrastive learning, which learns to compare two samples, we design the supervised classifier loss function L for classifying graphs as normal or vulnerable:

$$\begin{aligned} L &= -\sum_{n-1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n}} w_{y_n} x_{n,y_n} - \beta L_{pool}, \\ L_{pool} &= \frac{1}{n} \sum_i (t_i * \log(o_i) + (1 - t_i) * \log(1 - o_i)), \end{aligned} \quad (2.2)$$

where w_{y_n} is the weight of class labels to impose a high penalty when misclassifying the minority class. L_{pool} is graph pooling loss [123], β is a hyperparameter to balance the classification loss and pooling loss, x_{n,y_n} is the element of input embedding to the loss function, o_i is the logit output from each VIPool layer, and t_i is the binary label. In order to minimize false positives and false negatives, it is necessary to remove potential drifting samples and subsequently employ a classification model for detecting interactive threats. This approach can yield higher accuracy compared to solely relying on contrastive learning, as demonstrated in Table 2.6.

Cross-Domain Graph Transfer Learning

Another challenge is that the semantic knowledge from different platforms varies. Some platforms contain less semantic knowledge, making it hard to train a robust GNN model. For example, the interaction rules of apps from the SmartThings platform are limited, with only 135 apps suitable for analysis. However, a key insight is that the apps' descriptions are mainly about the functions of device controls on the smart home platforms, which share common information. For example, the app on the SmartThings platform "*Turn on the air conditioner when the temperature is above 85 °F*" and the skill on the Alexa platform "*Alexa, turn on the air conditioner*" trigger a similar action. Interaction knowledge can be transferred across different smart home platforms as these platforms all target the same trigger-action paradigm. Therefore, we design a transfer learning module to transfer knowledge of interaction information across platforms.

Transfer learning has demonstrated its superiority in enhancing model performance and reducing the training time with less training data in computer vision [80, 239]. Deep learning models typically show excellent transferability properties. For example, the first layer features tend not to be specific to a particular task [239]. Similarly, the first few layers of embedding features of GNN contain some common information. The node features are both semantic-based embeddings. In addition, they share some properties that are transferable among smart home platforms. For example, the knowledge in the IFTTT interaction graph can be transferred to the SmartThings interaction graph. In turn, the learned features of the SmartThings graph model can be used to train a better IFTTT graph model. We can transfer embedding features from other pre-trained models to train a more robust model.

Formally, suppose D_S is source domain and D_T is target domain, where $D = \{X, P(X)\}$ and

$X_S \neq X_T$. A domain feature space X is defined as a set of graph node features in a certain platform in the context of Glint. X_S is the feature space of the source domain, and X_T is the feature space of the target domain. The goal of transfer learning is to learn $P(Y_T|X_T)$ by gaining information from D_S and source task T_S. First, Glint trains a source domain model M_{T_S}. We take source domain graph node features as input to train M_{T_S}. Then, Glint transfers part of the layers of M_{T_S} to the target domain task model. Finally, Glint uses a small number of data in D_T to fine-tune parameters in the transferred layers and train parameters in new layers. The number of transferred layers depends on the specific graph model to yield better performance.

Glint uses a two-pronged approach for transfer learning. First, Glint transfers semantic knowledge from one platform to another, such as from the SmartThings interaction graph to the IFTTT interaction graph. There are common features in semantic knowledge from different platforms. Thus, Glint can transfer semantic knowledge in the form of feature embeddings. Second, Glint transfers interaction graph knowledge from homogeneous to heterogeneous interaction graph. Different platforms provide certain similar functions even though the usage is different. For example, an Alexa voice command or a SmartThings app can directly control the light status. The embeddings of GNN layers can transfer such interaction knowledge. Because the first several layers' features learned by GNNs are generic to be applied in many tasks [178], Glint can freeze the first several layers close to the input layer to preserve the transferred knowledge.

2.4: Evaluation and Discussion

We first evaluate the rule correlation discovery models for graph dataset construction. Then, we present and label the constructed interaction graphs. Next, for interactive threat analysis, we evaluate the performance of our designed ITGNN model on homogeneous and heterogeneous graph datasets. Then, we explore the possible interactive threats in user-designed automation rules. Finally, we evaluate Glint with a real-world testbed.

Table 2.2: The number of rules from 5 platforms.

IFTTT	SmartThings	Alexa Skill	Google Assistant	Home Assistant
316,928	185	5,506	5,292	574

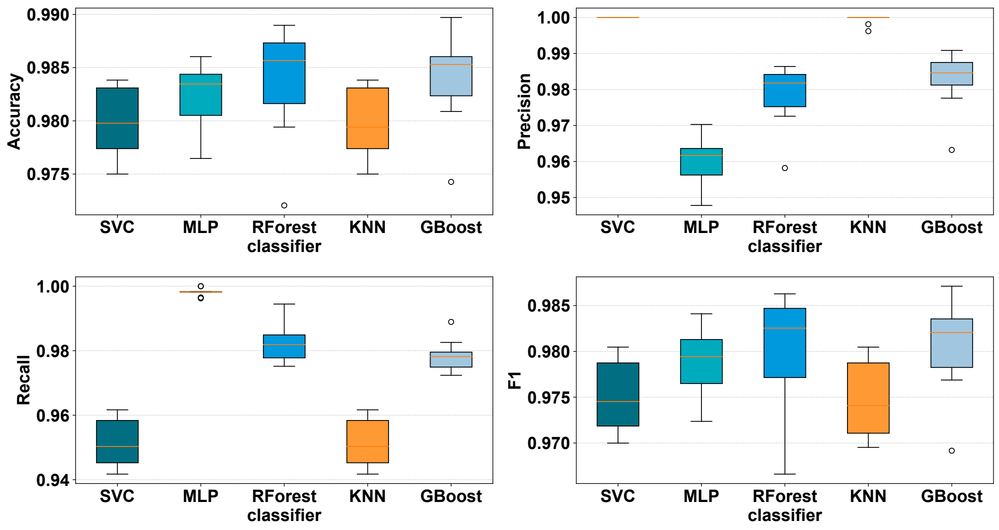


Figure 2.6: The performance of five classification models.

2.4.1: Rule Correlation Discovery Evaluation

We use Scrapy [181] to crawl publicly available rule descriptions from 5 smart home platforms, as shown in Table 2.2. We randomly choose and manually label 5,600 pairs that have action-trigger correlations and another 8,000 pairs that have no action-trigger correlations. Part of the labeled data is directly from [222], and we manually label the interaction pairs in newly crawled data. We use the labeled dataset to train classifiers to recognize whether a sentence pair has an action-trigger correlation. Then, we use the well-trained classifiers to label whether a sentence pair has action-trigger correlations for unlabeled rule pairs. Besides, we manually check the classification results to ensure the correctness of action-trigger correlations. In this way, we can efficiently filter out the impossible and unrealistic correlations among different rules.

We compare five different classification models: C-Support Vector (SVC), Multi-layer Perceptron (MLP), Random Forest (RForest), K-nearest Neighbors (KNN), and Gradient Boosting (GBoost) using Scikit-learn [164] library. The input of a model is the features of action-trigger pairs. The output is true or false. True means there is a correlation between “action” to “trigger”. To deal with the issue of class imbalance, we adjust class weights inversely proportional to class frequencies in the training data. For instance, we use a weight balance in the MLP loss function to give a high cost for the misclassification of “true” flows. We apply the grid search method to find the most effective hyperparameters and use 10-fold cross-validation to test the generalization of models. We compute the accuracy, recall, precision, and F1 to evaluate the models. The recall R represents the percentage of positive samples in the test set which are predicted correctly. A high recall means a low false-negative rate. The precision P shows the percentage of positive predictions which are truly positive. The high precision means a low false-positive rate. The F1 score is the harmonic average of precision and recall: $F1 = \frac{2 \cdot P \cdot R}{P+R}$.

Can ML aid in rule correlation discovery? As shown in Figure 2.6, all five models achieve excellent correlation classification performance, demonstrating the utility of the designed features. MLP and Random Forest achieve 98.2% and 98.4% accuracy, respectively. Both SVC and KNN achieve the highest average precision rates of 100%. The high precision ensures that the true connection predictions have a high possibility of being truly positive. MLP and Random Forest models achieve 99.8% and 98.2% recall, respectively. Besides, Random Forest achieves the highest 98% F1 scores. A higher F1 score for a classification model means better performance. With deep and large hidden layers, MLP can better characterize the hidden features in the dataset. The Random Forest model is based on bagged trees and uses a random subspace method to prevent overfitting. We chose MLP, RandomForest, and KNN to collaboratively predict the remaining 20,000 unlabeled pairs based on the highest precision, recall, and F1 value, so we can uncover the interaction correlations between actions and triggers. If the predictions of the three models differ, we manually review the prediction results to determine the final correlation labeling result.

2.4.2: Interaction Graph Construction

Due to the large volume of rule combinations, we randomly select and chain different rules that formulate “trigger-action” correlations to build interaction graphs, of which the number of nodes

Table 2.3: The number of interaction graph datasets.

Type	Platforms	Label	Num. of Total Graph	Num. of Unsafe Graph
Homo.	IFTTT	labeled	6,000	1,473
		unlabeled	10,000	*
	SmartThings	labeled	165	36
Hetero.	5 platforms	labeled	12,758	3,828
		unlabeled	19,440	*

is from 2 to 50. In an interaction graph, we build an edge between two rules that have a "trigger-action" correlation, and each rule is a node.

For the IFTTT interaction graph dataset construction, we use rules from both the dataset (315,393 applets) from [222] and our newly crawled 1,535 applets to construct interaction graphs among different applets. Considering descriptions could be lengthy and contain multiple sentences, we extract word phrases for text descriptions of IFTTT applets. Similarly, we build the SmartThings graph dataset by analyzing the apps' descriptions provided by the developers. For IFTTT and SmartThings homogeneous interaction graph datasets, we use the *en_core_web_lg* model in spaCy [195] to get the averaged word embeddings of each phrase in every rule. The dimension of embedding is 300, which is used as node features. As shown in Table 2.3, on the IFTTT platform, we build and manually check 6,000 interaction graphs to label whether graphs contain interactive threats or not. Finally, we label 1,473 graphs that contain interactive threats. Besides, we build 10,000 unlabeled IFTTT interaction graphs for discovering interactive threats in the smart home platform. On the SmartThings platform, we cross-check our 165 SmartThings inter-app interaction graphs with existing graphs [54] to ensure accuracy. We label 36 unsafe graphs out of 165 graphs.

We build a heterogeneous interaction graph dataset on IFTTT, SmartThings, Alexa, Google Assistant, and Home Assistant platforms by building interaction correlations across platforms. Specifically, we use IFTTT, SmartThings, and Alexa to build 12,758 heterogeneous interaction graphs with labels. We have heterogeneous graphs, among which 3,828 graphs are labeled as vulnerable graphs. For voice assistant rules, we use the *Universal Sentence Encoder* [29] to obtain sentence embeddings of rules, which are more suitable for identifying sentence features. The dimension of each embedding is 512, which is used as the node feature. Besides, Glint generates 19,440 unlabeled graphs based on the above five platforms. The Google Assistant and blueprints (rules) from Home Assistant are used in unlabeled graphs. These blueprints were created and discussed from December 2020 to January 2022.

We use DGL [221] to build the graph datasets. The stored labeled graph dataset file for IFTTT is 21.8G, the one for SmartThings is 0.018G, and the heterogeneous graph amounts to 81.6G. The large dataset file is partly due to the storage of all vertex information with DGL. We apply the DGL with Pytorch to build and train the GNN model. We run experiments on TensorEX Ubuntu 20.04 Deep Learning Stack with 256GB DDR4 memory, Intel(R) Xeon(R) Gold 5218R 2.10GHz CPUs and RTX A6000 GPUs.

In the labeling phase, two students who study IoT security volunteer to label interaction graphs. They first learn the security policies and IoT interactive threats identified in the literature [3, 28, 54, 55, 70, 203, 222]. For example, IoTSafe [55] specifies safety and security policies such as "All electrical appliances should be turned off when smoke is detected". Then, two volunteers follow the six types of threats [222] as criteria for labeling, where the specific rule settings are in Table 2.4:

(1) Condition bypass. Users may configure different granularity of settings with the same aim on different platforms. As a result, some conditions in fine-grained settings may be bypassed due to existing coarse-grained settings. For example, the users may have SmartThings setting 1 and have a concise setting 2 with Alexa. Such settings will allow condition bypass and cause threats.

(2) Condition block. With different smart home settings, the condition may be blocked because some commands may cancel the execution condition of another command. For instance, suppose we have settings 3, 4, and 5; when motion is detected at the door after 7 p.m., no notification is sent because the home state is disarmed.

(3) Action revert. The same device may be correlated to multiple environmental factors, and the action could be reverted if some environmental factors change. For example, with settings 6 and 7, the AC is turned on and then turned off. Such interactions may cause action to revert.

Table 2.4: Rule examples from different platforms.

Platform Setting	Content
SmartThings Setting 1	If outside temperature is above 70°F and time is 11 a.m., then open windows.
Alexa Setting 2	If outside temperature is above 70°F, then open windows.
IFTTT Setting 3	If motion is detected at the door and home is in armed state, then send a notification.
IFTTT Setting 4	When light is on, disarm home state.
SmartThings Setting 5	Turn on the light at 7 p.m.
Alexa Setting 6	Turn on the air conditioner when temperature is above 100°F.
IFTTT Setting 7	When humidity is below 30%, turn on humidifier and turn off air conditioner.
SmartThings Setting 8	If smoke is detected, unlock the door.
Alexa Setting 9	Lock the door at 10 p.m. every day
IFTTT Setting 10	Turn off the living-room light when bedroom light is on.
IFTTT Setting 11	If the living-room light is turned off and the home state is away, then turn on the bedroom light.
Alexa Setting 12	Turn on a heater
SmartThings Setting 13	Open windows if indoor temperature is above 80°F.

(4) Action conflict. Some actions may conflict with each other in multiple platforms. For instance, with settings 8 and 9, the action is conflicted if smoke is detected after 10 p.m.

(5) Action loop. The interactions among devices and the environment could trigger each other, which will introduce a loop in the interaction graph. For example, with settings 10 and 11, when the home state is away, the actions will loop forever.

(6) Goal conflict. Users could make misconfigurations, such that the actions' goals conflict with each other. Suppose the user turns on the heater to keep a room warm while in the bathroom; however, given the settings 12 and 13, the window opens, resulting in the temperature drop.

We develop the Python scripts to show the graph and rule contents, so the first volunteer can quickly check the graph edges and derive the conclusion. If a type of defined threat is found, then the volunteer stops checking and labels the graph as vulnerable. Otherwise, it is labeled as normal. Especially we cross-check our 165 SmartThings inter-app interaction graphs with the existing inter-app interaction chain graph of SmartThings [54] to ensure accuracy, from which we label 36 unsafe

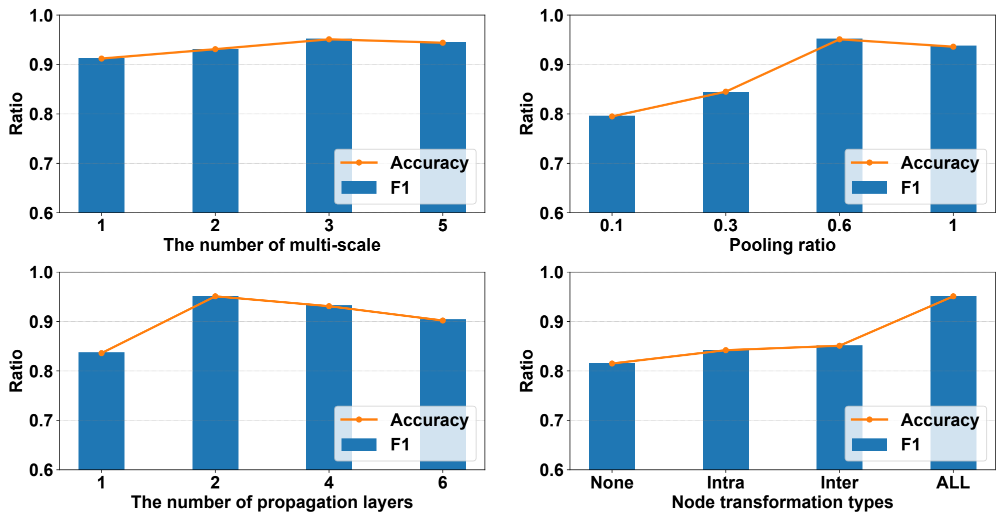


Figure 2.7: The ablation study of ITGNN model.

graphs out of 165 graphs. Note that there could be multiple types of interactive vulnerabilities in a graph, which can be future work if fine-grained labeling data is required.

The other volunteer double-checks the labeling result to avoid mislabeling. If they have disagreements on the labels, they will have a further discussion based on the security policies identified in the literature. For large and complex interaction graphs, such as a graph with 50 nodes, it takes about 6 minutes to examine the possible interactive threats in a graph. For simple graphs with two nodes, we just need to check whether they violate the security policies, which takes about 20 seconds. On average, it takes about 1 minute to label each graph at the beginning. After the two volunteers get familiar with the labeling process, it takes about 40 seconds to label and double-check each graph. In total, it takes about 8 weeks to label the entire graph dataset.

2.4.3: Ablation Study of ITGNN

In this section, we implement an ablation study focusing on the ITGNN model. We evaluate the hyper-parameters and components of ITGNN on the heterogeneous dataset as shown in Figure 2.7. Specifically, we consider (i) The number of scales, which is in the multi-scale graph generator. We can see that ITGNN achieves the best performance when the number of scales is 3 in Figure 2.7. When the number of scales is small (e.g., 1, original scale), ITGNN underestimates

Table 2.5: Accuracy for each type of vulnerability.

Condition bypass	Condition block	Action revert	Action conflict	Action loop	Goal conflict
94%	94%	95%	98%	93%	95%

local information in graphs while global information is underestimated when the scale number is large (e.g., 5). (ii) Pooling ratio, which is graph pooling in VIPool. When the value is 1, the model keeps all vertices and yields VIPool ineffective. When the value is 0.6, ITGNN can select vertices that can better express neighborhoods' information, which helps achieve the best performance.

(iii) The number of propagation layers in the multi-scale graph generator. We test the influence of the number of layers in the propagation process. As shown in Figure 2.7, when the number of layers becomes large (e.g., 6), the model achieves worse performance compared with the number of layers being 2. The GNN model suffers from over-smoothing and becomes less expressive when the layer number increases. (iv) We also remove some modules in metapath-based node transformation. As shown in Figure 2.7, when we remove both intra-meta path and inter-meta path aggregation modules, the model can only achieve an accuracy of 81.5%. For heterogeneous graphs, the model ignores the various types of node information in different metapath instances. Thus, the model cannot effectively learn the graph structure information in heterogeneous graphs. In contrast, the complete design of ITGNN achieves the best accuracy of 95.1%.

We also report accuracy for each type of vulnerability in Table 2.5. Considering a graph can contain multiple types of threats, our labeled dataset does not label the specific types of threats. Thus, for each vulnerability, we manually sample 50 graphs and mix them with 50 benign graphs to compose a test set. In total, there are 6 test sets, where each test set is corresponding to one vulnerability. From Table 2.5, we notice the action conflict is best recognized by Glint because it contains conspicuous antonym semantics.

Table 2.6: Results of homogeneous graph classification.

Dataset	Model	Accuracy	Precision	Recall	F1
IFTTT	GCN [108]	89.5	100	89.5	94.5
	GXN [123]	78.7	79.0	76.4	76.3
	GIN [231]	95	94.7	94	94.4
	IFG [196]	69.8	75.5	70.2	67.4
	SVC [166]	84.1	84.1	84	83.9
	KNN [4]	89.5	90.9	89.5	89.6
	ITGNN-C	95.4	95.3	94.9	95
	ITGNN-S	**95.7**	**95.9**	**95.7**	**95.8**
SmartThings	GCN [108]	90.9	82.6	90.9	86.6
	GXN [123]	88.2	89.9	88.2	87.2
	GIN [231]	89.7	85.9	89.5	87.7
	IFG [196]	86.1	89.3	87.5	85.9
	SVC [166]	84.4	87.3	84.8	81.3
	KNN [4]	84.8	83.8	84.8	83.2
	ITGNN-C	76.5	69	70.6	69.5
	ITGNN-S	**88.2**	**89.9**	**88.2**	**87.2**

2.4.4: Homogeneous Graph Evaluation

With the interaction graph datasets, we can train the GNN-based interactive threats detection model. We run experiments in 5 trials. For each trial, we split the dataset into the training set and testing set by 8:2. However, there are only 1,473 vulnerable interaction graphs out of 6,000 interaction graphs, as shown in Table 2.3. The imbalanced classes will degrade the performance of the model. Thus, for the training set, we first increase the number of vulnerable graphs by random oversampling until the number of vulnerable graphs is doubled. Second, we assign unequal costs to the misclassification class. The class weight is inversely proportional to the number of class examples. We use the weighted F1 score to measure the performance of graph models. We calculate the F1 value for each label and then compute the average weights by the number of occurrences of each class. Thus, the F1 value may be beyond the range of precision and recall.

We compare ITGNN with four state-of-the-art graph models and two classification models, including: (i) graph convolutional network (GCN) [108], which is a convolutional network operating on graphs; (ii) graph cross-network (GXN) [123] model, which extracts graph features by learning from multiple scales of a graph; (iii) GIN [231] model, a graph isomorphism network, which is used to distinguish certain graph structures; (iv) InfoGraph (IFG) [196], which uses graph-level representation learning via mutual information maximization. (v) Two classical ML methods: SVM and KNN. We compute the average of the node embeddings as the graph sample feature. (vi) ITGNN-C uses the contrastive learning loss in Eq. (2.1) and ITGNN-S uses the classification loss in Eq. (2.2) to differentiate between vulnerable and normal interaction graphs. We train the eight models and fine-tune hyperparameters on interaction graph datasets.

Why not use traditional ML models? As shown in Table 2.6, on the IFTTT dataset, the two traditional classifiers, SVC and KNN, achieve 84.1% and 89.5% accuracy, respectively. Nevertheless, the graph models can achieve better performance. For example, compared to KNN, GIN improves accuracy by 5.5% to 95%, and ITGNN-S improves accuracy by 6.2% to 95.7%. Traditional ML models cannot mine the graph structure information, while GNN is designed to learn graph information, and it can better mine the interactive threat patterns in graph structures.

Why not use the contrastive loss for classification? Compared with models trained on the IFTTT dataset, models trained on the SmartThings dataset have worse performance. For instance, ITGNN-C on the SmartThings dataset only achieves 76.5% accuracy because contrastive learning needs a large number of data to learn to compare two samples. However, the size of the SmartThings dataset (165 graphs) is much smaller than that of the IFTTT dataset.

2.4.5: Heterogeneous Graph Evaluation

We evaluate the performance of our proposed ITGNN model on the heterogeneous graph dataset. We compare ITGNN with three state-of-the-art heterogeneous graph learning methods. (i) HGSL [249] performs heterogeneous graph structure learning for classification. The MAGNN [71] is designed for heterogeneous node classification and link prediction. Due to the different downstream tasks in the original models, we construct two heterogeneous graph models. (ii) MAGCN model is an adapted GCN [108] model with a MAGNN graph converter. (iii) MAGXN model is then adapted GXN [123] model with a MAGNN graph converter.

Why not use existing GNN models? With the evaluations on both homogeneous and

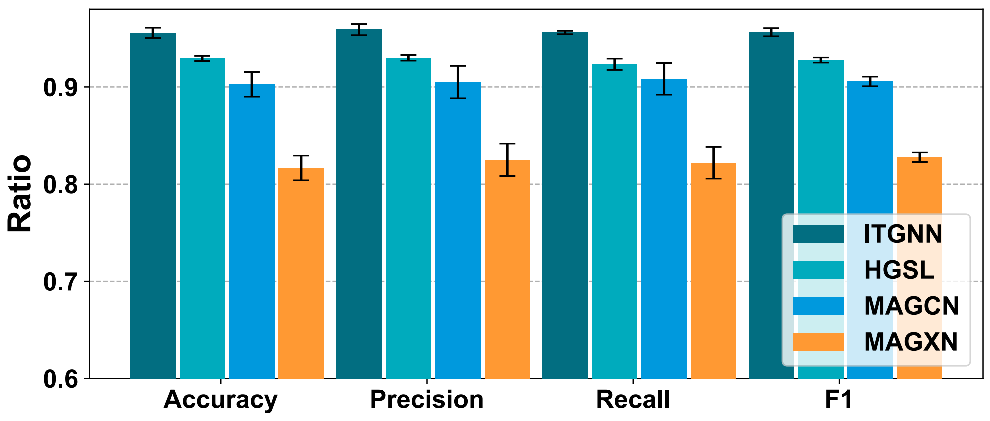


Figure 2.8: The results of heterogeneous graph classification.

heterogeneous graph datasets, we show the benefits of ITGNN as a unified framework for both homogeneous and heterogeneous graph representation learning. Besides, ITGNN is designed for both drifting sample detection and interactive threat classification problems. On the IFTTT dataset, as shown in Table 2.6, ITGNN achieves the best accuracy, recall, and F1 score. Specifically, ITGNN-S achieves 95.7% recall and F1 score on the IFTTT dataset. Achieving a better recall means that ITGNN-S avoids safety and security problems by better reducing false negatives of interactive threats. Even though GIN performs better on the SmartThings dataset, it achieves lower precision than ITGNN-S. Other complex GNN architectures entangle the whole interaction graph information, which is not suitable for our interactive threat detection task.

On the heterogeneous graph dataset, as shown in Figure 2.8, our proposed ITGNN model can achieve the best average accuracy (95.5%), precision (95.9%), recall (95.6%), and F1 score (95.6%). The HGSL achieves 92.9% accuracy, the MAGCN achieves 90.2% accuracy while MAGXN achieves 81.7% accuracy for threat detection. The MAGXN has a more complex architecture with more parameters, resulting in a slow training process. By contrast, the MAGCN achieves acceptable performance even though it has a simpler structure, which confirms the “No free lunch” theorem [227] for machine learning. Overall, the proposed ITGNN outperforms both the homogeneous and heterogeneous models. ITGNN is highly effective since it can not only preserve heterogeneous node feature information but also leverages the multi-scale graph structure features to generate graph representations.

2.4.6: Transfer Learning Evaluation

From previous results in Table 2.6, we find that the simple GCN architecture can be easily trained, while other models such as GXN and ITGNN have relatively complex architecture, which can be overfitting on an insufficient dataset (e.g., SmartThings). This indicates that one model may best perform for some datasets, but not in all cases. To apply Glint in a more diverse type of heterogeneous smart home platforms, we need to enhance the adaptability of GNN models. Transfer learning applies the learned knowledge from one domain to a different but related domain, improving modeling performance with less training data and time.

As shown in Table 2.7, we not only evaluate the transfer learning between homogeneous (IFTTT)-homogeneous (SmartThings) datasets but we also evaluate homogeneous (IFTTT)-heterogeneous datasets. We take IFTTT and SmartThings platforms as examples of homogeneous graph learning because these two are prevalent platforms in IoT interaction studies. Besides, the IFTTT dataset contains a large number of graphs, while SmartThings contains a small number of graphs, which are representative datasets for transfer learning problems. The size of the SmartThings dataset is much smaller than the IFTTT dataset. Therefore, given the IFTTT dataset as the source domain and the SmartThings dataset as the target domain, we only fine-tune the fully connected layer for classification and freeze the other layers of a model trained on the IFTTT dataset. Then, we let the SmartThings dataset be the source domain and the IFTTT dataset be the target domain; now, we can freeze the earlier two layers of the model trained on the SmartThings dataset. The earlier layers capture the basic features of interaction graphs. Finally, we can train the rest of the layers using the IFTTT dataset.

Can transfer learning improve model performance? The performance of all models improves with transfer learning techniques as shown in Table 2.7. For example, the accuracy of the GIN model on the SmartThings dataset improves by 6%. One interesting phenomenon is that if a model performs well in the source domain, it will also work well in the target domain. For instance, the ITGNN model achieves 88.2% accuracy on the target domain (SmartThings). With knowledge of the source domain (IFTTT), the model can achieve 100% accuracy. Moreover,

Table 2.7: Detailed performance of transfer learning.

Model	Target Domain	Source Domain	No trans.	Trans.	Improved
GIN	SmartThings	IFTTT	89.7%	92.3%	2.6%
GIN	IFTTT	SmartThings	95.0%	95.2%	0.2%
GCN	SmartThings	IFTTT	90.9%	94.1%	3.2%
GCN	IFTTT	SmartThings	89.5%	93.9%	4.4%
ITGNN	SmartThings	IFTTT	88.2%	100%	11.8%
ITGNN	IFTTT	SmartThings	95.7%	96.4%	0.7%
ITGNN	IFTTT	Heterogeneous	95.7%	96.1%	0.4%
ITGNN	Heterogeneous	IFTTT	95.1%	95.5%	0.4%

even though the source domain (SmartThings) knowledge is limited, it can still provide helpful knowledge across platforms. For example, the accuracy on the IFTTT platform grows from 95.7% to 96.4%. Moreover, we check if the transfer learning techniques could bring negative effects. As shown in Table 2.7, the applied transfer learning techniques consistently enhance the model by achieving better accuracy in the target domain. Therefore, models trained on datasets from multiple smart home platforms do not incur negative transfer problems.

2.4.7: Drifting Interaction Pattern Evaluation

We implement the Algorithm 3, and we use principal component analysis (PCA) to project the graph embeddings from the original 256-dimensional into a 2-dimensional space. We use the K-means method to cluster the projected embeddings as shown in Figure 2.9, where the centroid of each class is the white cross.

We test the samples in the unlabeled dataset to detect the potential drifting samples (the samples in the red circle in Figure 2.9). The drifting samples are examined manually to summarize the potential interaction patterns and can be used for retraining. Among 10,000 unlabeled IFTTT graphs, we found 63 potential drifting samples. For 19,440 unlabeled heterogeneous graphs, we found 104 potential drifting samples. For example, the opening and closing of a lawn sprinkler valve is a rare example and only exists in the unlabeled dataset, so it is regarded as a potential drifting sample. The ratios of vulnerable interaction graphs on the randomly generated unlabeled IFTTT dataset and heterogeneous dataset are 8.3% and 16.7%, respectively. To recognize whether

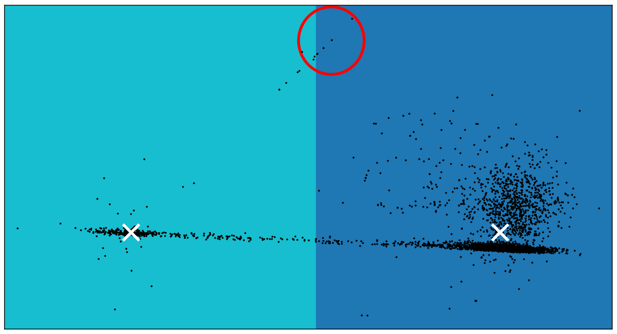


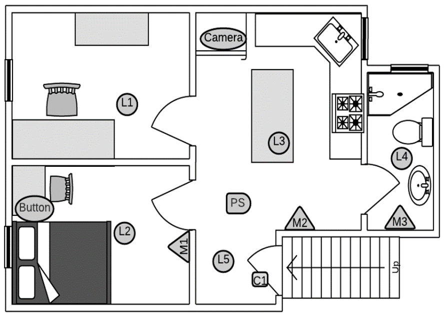
Figure 2.9: K-means clustering on graph embeddings learned with contrastive loss.

there are new threat cases, we manually examine all drifting samples and randomly check 50% predicted vulnerable interaction graphs. Primarily, in drifting samples, we discover four new types of interactive threats related to user-designed blueprints (rules) [10] across multiple platforms. The reported new interactive threat types are more complex and easily overlooked, while the previously discovered threat types from existing work are all rooted in a single platform.

• **Action block**: Automation is blocked by non-automation settings. Users create automation blocker rules that block some automation from running. For example, users can set rule 1, "If the light is set in manual mode, then keep the light brightness to 100%.". Thus, rule 2, "dimming lights when turning on the TV" will become ineffective. Rules 1 and 2 are encoded into two nodes in an interaction graph, and they target the same device, "light", which will be labeled as a vulnerable interaction because of the block of the action.

• **Action ablation**: Automation action can be reverted over time. A device can have multiple attributes that are triggered by different factors. For example, a blueprint turns on the air conditioner (AC) when the temperature is above 95°F. Another blueprint has the rule that turns on the humidifier and turns off the AC when humidity is below 30%. The status of the AC between the two rules is conflicted and will be reverted when the humidity becomes low.

• **Trigger intake**: Automation is caused by unexpected triggers. A user applies a rule to send the camera a snapshot notification when motion is detected at the door. There is another rule that starts the vacuum cleaner at 9 pm, which could accidentally trigger the motion sensor. As a result, the user may frequently receive false snapshot notifications. Without a deep analysis, the user can hardly



Abbr.	Device Name	Attributes
L	Light Bulb	switch
M	Motion Sensor	motion
C	Contact Sensor	contact
T	Temperature Sensor	Temperature
PS	Presence Sensor	presence
Camera	Camera	snapshot
Button	Smart Button	button

Figure 2.10: The layout of smart home devices.

notice this is caused by the event that the vacuum cleaner triggers the motion sensor, especially in a bad light condition.

• **Condition duplicate**: A fake automation condition is generated by rules from other platforms. For instance, a rule that reports the room is occupied when any of the conditions are met: motion sensor detects motion, the door is shut, or media is playing on devices in the room. A rule from IFTTT can play music in the room from 3 pm to 4 pm. Due to the trigger from the played music, the room is reported to be occupied. Another rule that starts the heating when the room is occupied and when the temperature is below 60°F is subsequently triggered. Because of the applet from IFTTT, the rule condition is accidentally satisfied in the platform of Blueprint.

Note that action block involves user intervention and action ablation happens over a long time. They are neither action revert nor action conflict defined in iRuler [222]. Besides, existing work such as HAWatcher [70] cannot handle these user interventions and long-term correlations. Similarly, trigger intake and condition duplicates are not pre-defined in iRuler or mined in HAWatcher. As a result, the existing work cannot detect these new types of interactive threats.

2.4.8: Evaluation on a Real-life Testbed

We construct real-time interaction graphs with event logs collected from a real-world home testbed, as shown in Figure 2.10. A volunteer deploys the SmartThings and Alexa systems in his house following the setup in HAWatcher [70] with off-the-shelf devices and apps. The volunteer uses the desired automation and spends a week collecting 1,813 event logs related to home automation.

Efficacy Comparison

Following the setup in HAWatcher [70], we simulate five types of attacks by modifying event logs or manually interfering with the home automation: (i) Targeted compromise: fake commands, stealthy commands. (ii) Interaction abuse: fake events, event losses. (iii) Misconfiguration: command failure. For example, we simulate “fake commands” by manually turning off lights during normal operation, and “stealthy commands” by manually starting a robot vacuum to trigger motion sensors. We build 600 graphs as the test set. Out of 300 graphs, 150 graphs contain binary-correlation threats (BCT). Among another 300 graphs, 150 graphs contain complex-correlation threats (CCT). BCT means interactive threats are caused by two nodes, while CCT is caused by more than two nodes.

We choose three white-box anomaly detection methods for comparison: **(i)** HAWatcher [70], which extracts binary correlations and verifies them with run-time event logs. The input is the device states extracted from event logs. The inconsistencies are reported as anomalies. We reuse the refined correlations reported in the work [70] to check the actual states of devices. **(ii)** One-class Support Vector Machine (OCSVM) [180], which is a unsupervised anomaly detection method. **(iii)** IsolationForest [131], which is also widely used for anomaly detection. We capture all devices’ states as a frame when a new event happens. Four consecutive frames compose a data vector, which is the input of OCSVM and IsolationForest. The output is -1 for threats and 1 for normal cases. We use Scikit-learn [164] to implement OCSVM and IsolationForest. Note that for many other systems, it is unfair to directly compare the performance of existing methods and ours when considering the big gap between the analysis of open-source and closed-source platforms.

As shown in Figure 2.11, OCSVM and IsolationForest achieve worse performance compared to Glint. For instance, for complex graphs, the OCSVM can only achieve 66.9% precision and 63.3% recall. This is because they only use time-series event log data, without the consideration of rich information in neighbor nodes and graph structure. The HAWatcher can achieve 97.8% precision and 94.1% recall taking 21 days of training for binary-correlation threat detection [70]. By contrast, Glint takes no more than 1 hour to train the model and apply transfer learning to improve model

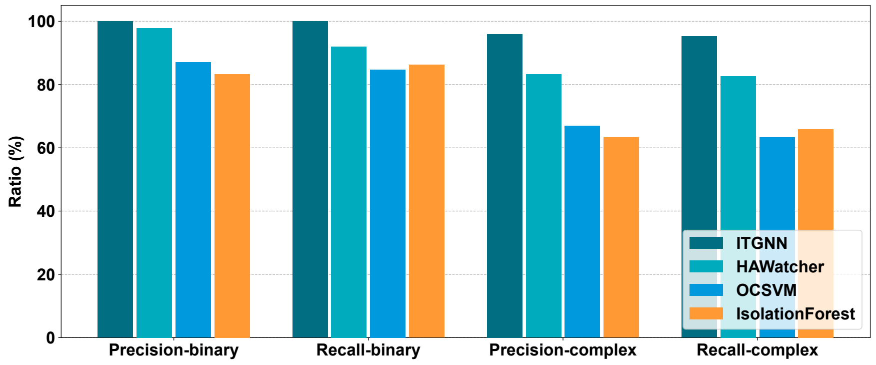


Figure 2.11: Comparison with different detection methods.

performance, which achieves 100% precision and recall for binary-correlation threat detection. For complex-correlation graphs, HAWatcher cannot check interactive threats caused by goal conflict, action revert, and condition bypass because the three types of threats are not covered. For such cases, we set HAWatcher to generate results by the Bernoulli distribution with the probability of a single success of 0.5. Finally, HAWatcher achieves 83.2% precision and 82.7% recall. By contrast, Glint achieves 96% precision and 95.3% recall, which outperforms HAWatcher in heterogeneous platform scenarios by 12.8% precision and 12.6% recall rate.

Efficiency Comparison

The efficiency is affected by many factors, such as code implementation, third-party libraries, and hardware setup, we qualitatively compare the efficiency of different systems. HAWatcher [70] needs to traverse all defined correlations. Suppose the number of correlations is n, then the time complexity is n^2 for their extracted binary correlations. The time complexity will be n^N when the length of the interaction chain is N. Thus, such search-based methods are retarded by the path explosion when dealing with large and complex graphs. iRuler [222] applies satisfiability modulo theories (SMT) solver to check interactive vulnerabilities. However, the existence of heterogeneous platforms indicates that there will be more complex rules for smart home management. With the fact that the SMT problem is typically NP-hard, such symbolic execution has poor scalability when dealing with large and complex interaction graphs. It is thus unlikely that a symbolic execution engine can exhaustively explore all the possible states within a reasonable amount of time. By contrast, our proposed Glint is a learning-based system, which achieves a high efficiency for real-time interactive threat prediction. For each graph, the prediction time is related to the graph size, and the average prediction for a heterogeneous graph takes about 0.61s. The ITGNN model trained on heterogeneous graphs is only 6.13 MB.

2.4.9: Discussion

While the above evaluation results are encouraging, we consider this work as a first step toward data-centric AI-based IoT interactive threat analysis. There are many factors that need to be considered in the future work.

Time factor

The IoT device influence can occur in the long term. For example, turning on/off the heater will change the temperature over a relatively long period. The existing system, HAWatcher [70], only considers interactions that arise within a short time frame, which cannot mine long-term correlations. By contrast, Glint discovers interactions based on automation rule semantics and real-time event logs. Glint encodes the time-related semantics into node embeddings. By annotating such long-term correlations in graph datasets, the GNN models can effectively learn such patterns manifested over a long time interval. For example, settings 12 and 13 in Table 2.4 compose a vulnerable interaction graph, which can be discovered by Glint but not by HAWatcher.

User factor

The automatic interactions could be disturbed by user-activity deviations. An example is shown in the drifting interaction pattern “Action block” in Section 2.4.7. Existing systems such as iRuler [222] use pre-defined vulnerability patterns, which cannot report such interactive threats. In comparison, Glint can detect abnormal patterns with concept drift detection. According to users’ feedback, Glint can fine-tune the model and adapt to user preferences. How regularly we should update the interaction graphs to accommodate changes in user actions and IoT devices poses an important future research problem.

NLP techniques

In this dissertation, we rely on the NLP techniques to identify the correlations between different automation configuration rules, so as to construct an automation rule interaction graph. However, the accuracy of interaction graph construction will affect threat detection results because of the inaccurate input to the threat detection model. Applying NLP techniques may overestimate or underestimate physical channel properties, especially when involving different locations. The physical properties of two devices at different locations can vary. For instance, the temperature of the oven in the kitchen can hardly influence the temperature in the living room. As a result, the lack of complete understanding of the physical channel properties can lead to wrong predictions about the correlation between the two rules. For example, "Set the temperature of the oven to 350°F for preheating at 7:00 pm" should have no correlation to "Open the window if the indoor temperature is above 80°F". A potential solution is to request feedback on the correctness of the built graphs from users, so we can fine-tune the graph builder model. Besides, more advanced NLP techniques such as large language models [157, 210, 255] can be applied to enhance the accuracy of IoT rule correlation recognition.

2.5: Summary

To detect interactive threats across heterogeneous closed-source platforms, we propose the Glint to learn interaction patterns. We design the unified ITGNN model for multi-scale graph representation learning. To train ITGNN, we build the first interaction graph dataset on five high-profile platforms. Our evaluation shows that the proposed model can achieve high accuracy in detecting interactive threats across heterogeneous platforms. We identify four new types of interactive threats in the user-defined Home Assistant platform. However, there are two remaining issues. Firstly, the training data for threat detection incorporates event log details, encapsulating sensitive information about daily activities. Consequently, the training data should be avoided sharing with the third party considering the privacy concerns. Secondly, there is a necessity to delve into the explainability of GNN models, which is crucial for unraveling the underlying root causes behind interactive threats.

CHAPTER 3: FEDERATED THREAT DETECTION MODEL LEARNING[2]

3.1: Introduction

As introduced in Chapter 2, a growing number of smart devices are deployed in modern homes to achieve home automation. The devices are controlled by rules that follow the trigger-action paradigm. For example, a SmartThings [190] app has the automation rule (R1) *"If smoke is detected (trigger), turn on the water valve and start alarm beeping (action)"*. According to a recent survey [38], 82.4% of smart homes have multiple rules for controlling a single device. These rules allow devices to interact with each other, referred to as IoT interaction.

However, unexpected IoT interactions could lead to threats such as action conflict and action revert, which result in severe security and privacy risks. For instance, suppose a user has set up the above SmartThings rule R1. Unexpectedly, as soon as the water valve was turned on and a water leak was detected in the kitchen, the SmartThings rule R2 *"Close the water valve when a water leak is detected"* would close the water valve. In consequence, the two rules compose a vulnerable interaction, which exposes the threat "*the water valve fails to turn on when smoke is detected*" because of the action conflict "*water valve opening and closing*". Such vulnerable interactions can be caused by user errors [93, 153, 205] and physical attacks [82, 224, 234, 245].

It is crucial to manage the IoT automation rules and track the trigger-action information flow to avoid interactive threats. Naturally, the IoT automation rules follow the general trigger-action paradigm, and the interactions among rules can compose an interaction graph. Automation rules

[2]This chapter is based on previously published work by Guangjing Wang, Hanqing Guo, Anran Li, Xiaorui Liu, and Qiben Yan titled "Federated IoT Interaction Vulnerability Analysis" published in the Proceedings of IEEE 39th International Conference on Data Engineering (ICDE). DOI: 10.1109/ICDE55515.2023.00120 [214]. © 2023 IEEE. Reprinted, with permission, from Guangjing Wang, Hanqing Guo, Anran Li, Xiaorui Liu, and Qiben Yan, "Federated IoT Interaction Vulnerability Analysis" (paper and IEEE titles are the same), July 2023.

can be represented by nodes, and the edges are "trigger-action" connections among different rules. A naive idea is to detect the sensitive event based on predefined security policies, and then trace it back to find root causes. However, the query or search-based methods have limited coverage of the security policies or threat patterns. Most methods [26, 28, 55, 70, 223] pre-define security policies and threat patterns within a single platform [38]. For example, ProvThings [223] requires users to define and input policies describing sequences of causal interactions with a graph database backend. Yet, the predefined policies can hardly cover potential new threats across heterogeneous platforms. As a result, it may yield significant false positive and false negative errors. Moreover, federated query-based methods [15, 85, 107, 202] are not applicable. Even though common security policies could be extracted and shared among multiple houses, device interactions in different houses do not intersect. Each interaction graph is a complete data sample. Therefore, the query outcome for threat detection in a single house cannot necessarily be computed over the union of source databases of multiple houses.

Another approach profiles the behaviors of systems by analyzing event logs [59, 127, 257]. However, they cannot fully expose the interaction correlations between different events. Their limitations are three-fold. First, considering the complex causal dependencies among multiple automation rules, it is hard to accurately mine the cross-app interaction logic from event log sequences, which are within individual devices [70]. Second, sharing smart home usage data with a third party might cause privacy leakage issues such as the leakage of living habits and routines, while a dataset from a single house is insufficient for training a model with high generalization ability. Third, the mining results are difficult to interpret. When a threat alarm is generated, the users can hardly learn the exact segment of device interactions that caused the issue.

To address these problems and limitations, we propose **FexIoT**, a **F**ederated and **ex**plicable GNN-based approach for automatic **IoT** interactive threat analysis. For an interaction graph, the node features can be represented by semantic-aware word or sentence embeddings. Compared with benign graphs, vulnerable interaction graphs could have different graph patterns that express abnormal behaviors. FexIoT utilizes the GNN model to learn the interaction patterns. Meanwhile, as the event logs belong to different platforms and households, by leveraging federated learning, we can collaboratively develop a more generalized model while retaining users' data locally. Moreover, the interaction graph allows to search and extract a special subgraph that contains execution flow paths related to abnormal behaviors. Based on the extracted flow path, we can explain the model's predictions and identify the causes of vulnerable interactions. There are two key technical challenges for IoT interaction analysis across multiple closed-source platforms.

(i) How to build the threat detection model with high heterogeneity in datasets while avoiding sharing raw data? One household has a limited number of devices, which makes it infeasible to train a custom and robust GNN model. FL allows for collaboratively training a model without exposing raw data. However, the data heterogeneity leads to poor modeling performance with slow convergence and low accuracy in FL training paradigms [9, 45, 199], especially when the data distribution is time-varying or not well balanced among different clients. To overcome this challenge, we design a layer-wise clustering-based federated graph contrastive learning framework. We train a shared threat detection model by differentiating normal and vulnerable graphs over *non-i.i.d.* datasets with high generalization ability without sharing users' data.

(ii) How to adapt to new threat patterns and automatically identify potential causes of vulnerable interactions? The interactions in the IoT world are complex due to the different types of devices and threats. Even though we apply FL to train on various graph data from different home sources, we acknowledge that there could be drifting samples that evolve from existing threats or are novel kinds of threats. We propose to filter out drifting samples based on federated contrastive graph representations and the corresponding median absolute deviation. Then, we design an efficient cause analysis method by exploring different subgraphs of an interaction graph with Monte Carlo beam search (MCBS) [25], which is a heuristic game tree search algorithm. Based on IoT devices' potential correlation features, we propose using the optimal SHAP values [136] to measure the risk of subgraphs of a complete interaction graph. Thus, we can trace the most possible information flow chain relevant to the vulnerable interactions.

In summary, we make the following contributions:

- We design a layer-wise clustering-based federated contrastive GNN model for *non-i.i.d.* graph datasets to learn contrastive graph representations, and it achieves more than 90% average accuracy in threat detection.
- We propose an efficient cause analysis method based on Monte Carlo beam search and the SHAP value to evaluate the risk of subgraphs, which automatically provides explanations for any complex vulnerable interactions.
- We implement various experiments on large-scale datasets, which shows that our method outperforms state-of-the-art baselines with high accuracy and reduces communication costs.

3.2: Problem Definition

Interactive threats refer to the vulnerability coming from interactions between devices, and the environment. Specifically, we label the graph as vulnerable if it satisfies at least one of the 6 types of vulnerabilities identified by existing work [222]: *condition bypass, condition block, action revert, action loop, action conflict, and action duplicate.* Internal graph vulnerability refers to the vulnerabilities inherently from interaction graphs, while external graph vulnerability refers to the vulnerabilities caused by external attacks.

Instead of enumerating all possible interaction vulnerabilities, we design a federated GNN model to learn the vulnerability patterns. Given the interaction graph G, our task is to learn the graph embedding Z^t at each time t, and Z^t is used for vulnerable interaction prediction. Considering a graph can contain multiple types of vulnerabilities, multi-class classification is not suitable. Thus, the prediction problem is formulated as a binary classification problem $f(Z^t) \rightarrow y^t$ that maps the interaction graph embedding Z^t to the binary label y^t of interaction incident at a given time t. Finally, we identify a subgraph G_{sub} with the highest risk score by exploring various subgraphs with the SHAP-based Monte Carlo beam search, which explains the result of prediction y^t.

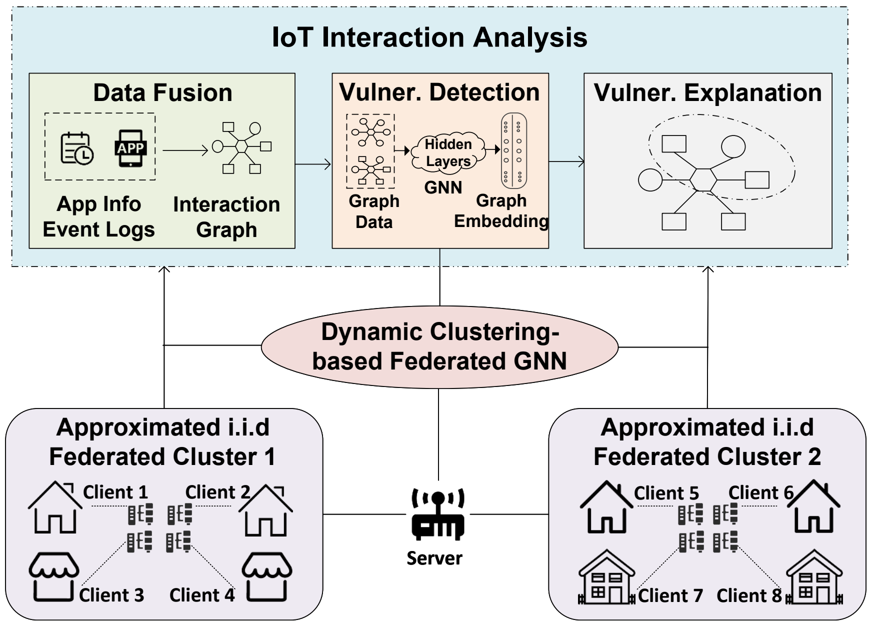


Figure 3.1: System architecture of FexIoT.

3.3: System Design

We design FexIoT for IoT interaction analysis as shown in Figure 3.1. A client represents a house where data is collected and a GNN model is trained using federated learning. Each client can run FexIoT on devices such as a Raspberry Pi or NVIDIA Jetson Nano. Each client implements three main components: data fusion from event logs and apps' descriptions, vulnerable interaction detection with federated GNN, and vulnerability explanation with the Monte-Carlo beam search-based method. A server can perform the clustering and aggregation in FexIoT, which could be served by a security solution provider.

3.3.1: Fine-grained Clustering-based Federated Graph Learning

After the graph construction, we propose a fine-grained clustering-based federated contrastive graph representation learning model considering data heterogeneity and drifting samples in FL. The learned graph representations can be used for vulnerability detection. In our designed FL paradigm, each client reserves two models. The first is the graph representation learning model, which participates in the FL process in Algorithm 4. The second is a linear classification model such as an SGDClassifier, which locally learns to classify the learned graph representation to detect vulnerable or normal graphs.

Contrastive Graph Learning Loss Function

In the graph representation learning model, we design the contrastive learning [35] loss function in Eq. (3.1) for learning a distance function to measure the dissimilarity of samples:

$$L_c = d_{ij}^2(1 - y_{ij}) + max(0, k - d_{ij}^2)y_{ij}, \tag{3.1}$$

where d_{ij} is the Euclidean distance between two graph embeddings, y_{ij} is 1 if graph G_i and G_j are from different classes, and y_{ij} is 0 if graph G_i and G_j are from the same class. We set a threshold k to restrict the unusual distance contribution of graphs from different classes. Note that we apply existing well-developed GNN models [71, 108, 231] in our federated learning framework. Finally, the learned representations are used for training a linear classification model to classify normal or vulnerable graphs in each client.

Dynamic Clustering-based Federated Learning

We design a fine-grained dynamic clustering-based FL algorithm as shown in 4. There could be domain shifts when the knowledge transfers across *non-i.i.d.* datasets in FL framework [43]. The heterogeneity is two-fold. First, the interaction graph can be either homogeneous or heterogeneous. Different households deploy heterogeneous devices and different users have their own usage habitats, which can cause graph heterogeneity. The heterogeneous graph data will cause negative transfer among users and decrease the model performance. Second, the graph dataset is unbalanced and *non-i.i.d.*. The *non-i.i.d.* graph datasets from different households will cause biased stochastic gradients, which will impede the convergence of FL models.

Another key observation is that despite the heterogeneity, there exist similar data distributions in smart homes because different users can have common automation rules. We assume that there exist several clusters of households, where the graph datasets from each cluster satisfy the *i.i.d.* property following [179]. We argue that the assumption is reasonable since even though there

Algorithm 4: 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($l$, $C$)
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
```

are heterogeneous smart devices, from the semantic perspective, the functionalities of devices are indeed limited. There will be some graph datasets that share common feature information, which could be grouped into several clusters. As a result, a new challenge ensues on how to effectively aggregate clients into different clusters.

Consider the clustering-based FL setting with a central server and a set of n clients $\{c_1, c_2, \cdots, c_n\}$, which can be dynamically clustered into different clusters $\{cluster_1, cluster_2, \cdots\}$. Each client c_i has a set of interaction graphs $G = \{G_1, G_2, \cdots\}$, and conducts the graph classification $y = h^*_k(G_i)$, where h^*_k is the optimal graph classification model for cluster set $cluster_k$. The graph feature information can be reflected by the model parameters and their gradients [179]. Existing federated graph classification over *non-i.i.d.* graphs [229] is coarse-grained since it only considers the similarity of parameters of a whole model. However, from the bottom up, the degree of similarity among deep models decreases [135, 151, 239]. Therefore, to learn the fine-grained clustering structure, we design the bottom-up layer-wise dynamic clustering algorithm to obtain the similarity among weights of clients as shown in Algorithm 4.

Suppose n is the number of clients in FL training, each client c performs local GNN training (lines 2-4), which follows the traditional FL training paradigm. The server receives n local models W_c^l (line 12). The dynamic clustering on the server starts from the bottom layer l_1. We define two thresholds ϵ_1 and ϵ_2 to determine the conditions of clustering of different clients (line 13):

$$\begin{aligned} &\epsilon_1 > || \sum_{i \in [n]} \frac{|G_{c_i}|}{|G|} \Delta W_{c_i} ||, \\ &\epsilon_2 < max(||\Delta W_{c_i}||), \end{aligned} \tag{3.2}$$

where $|G_i|$ represents the number of graphs owned by client i, $|G|$ is the total number of graphs owned by all clients, and ΔW_{c_i} is the local update of model weights of client c_i. ϵ_1 measures the degree of fluctuation of FL training, and it bounds the relatively stationary point of the global model before initiating the clustering. Meanwhile, if there are large norms of weight update that are greater than ϵ_2, it means high heterogeneity occurs among different clients, and the clustering starts to avoid performance degradation among clients. The two thresholds can be determined via initial experiments on the validation sets [179]. If the conditions in Eq. (3.2) are satisfied, the server further divides the clients from the same cluster into two sub-clusters and performs model aggregation within each sub-cluster (lines 14-17). This process continues to the next layer recursively. Thus, with more layers of client models being clustered and aggregated, each cluster of clients has reduced divergence, which achieves model converging in fewer training rounds.

Drifting Interaction Pattern Detection

Due to dynamic changes in smart device settings and the presence of different attacks, the testing interaction graph distribution may diverge from that of the training dataset in actual deployment scenarios. The performance of the detection may be impacted by the drifting samples, which are new interaction vulnerability patterns that are different from already known vulnerabilities. Considering the high false-positive or false-negative rate caused by drifting samples, we design the interaction graph drifting pattern analysis method based on the federated graph representation learning and median absolute deviation (MAD) [114] in the model application stage.

First, in each client, we use the well-trained model in FL to map all the training graph data into latent space. We can calculate the centroid of each class by computing the mean value for each dimension of the latent embedding. Similarly, for the incoming graph data, we also generate latent embeddings. In this way, we can compute the Euclidean distance d_i^k between the centroids of each class in the training dataset and the test sample x^k. Second, based on MAD, we can estimate the graph data distribution within each class i by computing MAD_i in the training dataset, which is the median of the absolute deviation from the median $\tilde{d}_i$ of distance d_i^j. The d_i^j is the distance between the latent embedding of each training sample to its centroid in class i. Third, for each testing sample, we check if d_i^k is large enough to make x^k an outlier of class i. Specifically, we denote the "normal" label as 0 and the "vulnerable" label as 1. Then $A_0^k = \frac{|d_i^k - \tilde{d}_i|}{MAD_0}$, $A_1^k = \frac{|d_i^k - \tilde{d}_i|}{MAD_1}$ and $A^k = min(A_0^k, A_1^k)$. If A^k is greater than a threshold T_M that is set as 3 empirically following existing practices [114], then x_t^k is a potential drifting sample. The MAD method allows each class to decide outliers based on its in-class distribution. In the model application stage, given a set of testing interaction graphs $\{G\}$, we first check each graph G_i to see if it is a drifting sample. If a new sample has a larger distance from all existing classes, then it is a potential drifting sample. Thus, we can filter out and manually inspect drifting samples that are outside of the training space.

3.3.2: Vulnerable Interaction Cause Analysis

Given a detected vulnerable graph, we design the SHAP-based Monte Carlo beam search (MCBS) algorithm to discover the causes of vulnerable interactions considering the IoT dependency relationship. We first formalize the vulnerability explanation problem. G represents the interaction graph that is examined by a GNN model followed by a linear classifier $h(\cdot)$ in the model application stage, and the prediction result is y. The vulnerable interactions are caused by a connected subgraph

G_{sub} in an interaction graph. We further define a cooperative game following [111] to measure the contributions of different parts (players) of a graph. Suppose G_{sub} is one player, where G_{sub} contains nodes $\{v_1, v_2, \cdots, v_s\}$. Other nodes in $G \backslash G_{sub}$ are other players $\{\{v_{s+1}\}, \cdots, \{v_m\}\}$. Our goal is to find the most possible subgraph G_{sub} that is responsible for the prediction y.

However, the device's abnormal behaviors are hardly noticeable until certain consequences occur. Moreover, an app can trigger unexpected events that are subscribed to by another device. Different apps could introduce contradictory changes to a device attribute. Directly using the prediction scores to measure the risk of subgraphs is problematic, since it cannot capture connections among different graph structures. Suppose there is a subgraph G_{sub} that contains the exact interaction vulnerability chain, any subgraph G_{con} containing G_{sub} will also cause the vulnerable interaction. Meanwhile, a subgraph G_{inc} that is included in G_{sub} may also trigger the same threat, but G_{inc} can only reveal part of the causes instead of the complete causal route. Another method SubgraphX [241] applies the Shapley value to measure the risk of subgraphs, where they assume the players are independent. However, in an interaction graph, the existence and actions of nodes are not independent because the deployment of devices is related to each other. Therefore, the Shapley value could neglect the dependent relations among different nodes.

We propose to calculate the SHapley Additive exPlanations [136] (SHAP) value to evaluate the risk score of subgraphs and apply MCBS to improve the efficiency of the search process as shown in Algorithm 5. In the MCBS tree, following the notation in SubgraphX [241], the root node N_0 is the input graph G, and each node N_i in the search tree is a connected subgraph G_{sub}, the edge in the search tree is the pruning action a. A subgraph G_j is acquired by the action a_j from G_i, which is represented by (N_i, a_j). We combine beam search with Monte Carlo tree search to store a set of best playouts, which is called *beam* (lines 2-4 in Algorithm 5). The size of a beam B_{level} is fixed for each level, which shows that B_{level} best nodes are kept at each level. From the model perspective, the target node feature is aggregated from a limited number of neighbor nodes. From the smart home perspective, devices are directly related to limited neighboring devices. Therefore, it is reasonable that only neighboring nodes in a beam B_{level} are employed for information aggregation.

Algorithm 5: SHAP based Monte Carlo Beam Search

```
Input: GNN model $h(\cdot)$, interaction graph $G$, MCBS iteration number $I$, kernel SHAP samples
       $K$, the least node number $N_{min}$, the root of search tree $N_0$
Output: subgraph $G_{sub}$
for $i < I$ do
    $S_i = N_0$, curPath=$\{N_0\}$
    while $|h(S_i)| > N_{min}$ do
        for subgraph $G_i$ in $B_{level}(S_i)$ do
            for $k < K$ do
                $g(z_r) = h(S_i)$
                $h(T_x^{-1}(z_r)) = h(S_i \bigcup G_i)$
            end
            $Score(h(\cdot, G, G_i)) = W$
        end
        Select $N_{next}$ following Eq. 3.7
        $S_i = N_{next}$
        $curPath = curPath + N_{next}$
    end
    $S_l = S_l \bigcup S_i$
end
return subgraph $G_{sub}$ with the highest score from $S_l$
```

The SHAP approach can rate the relevance of each feature for a specific prediction. SHAP values are the Shapley values of a conditional expectation function [136], which can better measure the feature importance. Here, we use $\{G_1, \cdots, G_i, \cdots, G_n\}$ to represent n subgraphs of an interaction graph G. The analysis result can be written as:

$$G^* = \underset{|G_i| \leq N_{min}}{\arg\max}\, \text{SHAP}(h(\cdot), G, G_i), \tag{3.3}$$

where N_{min} is a hyperparameter that limits the number of nodes in a subgraph. Following SHAP framework, the SHAP value ϕ_i can be computed as follows:

$$\phi_i(h, x) = \sum_{z_r \subseteq x_r} \frac{|z_r|!(M - |z_r| - 1)!}{M!}[h_x(z_r) - h_x(z_r \backslash i)], \tag{3.4}$$

where x_r is the set of total non-zero entries, z_r is a subset of x_r, $|z_r|$ is the number of non-zero entries in z_r, M is the number of set of all entries, $z_r \backslash i$ denotes setting $z_{i_r} = 0$. In our proposed

Algorithm 5, non-zero entries mean the nodes (players) of a graph that are included in the Monte Carlo beam search process. $h_x(z_r) = E[h(z)|z_S]$ means it calculates the expectation across all possible node combinations, where S is the non-zero index in z_r. Solving Eq. (3.4) is challenging. Thus, we apply the kernel SHAP to approximate it (lines 5-9 in Algorithm 5).

$$L(h, g) = \sum_{z_r \in Z} [h(T_z^{-1}(z_r)) - g(z_r)]^2 \cdot C, \tag{3.5}$$

$$C = \frac{M - 1}{\binom{M}{|z_r|} |z_r| (M - |z_r|)} \tag{3.6}$$

where we assume $g(z_r) = Wz_r$ is the explanation model following a linear form. $g(z_r)$ matches the original model $h(z)$ when $z = T_z(z_r)$. $T_z(\cdot)$ is a mapping function that converts the inputs to the original input space. We use the weighted linear regression to solve the equation since $L(h, g)$ is a squared loss and $g(z_r)$ is the linear function. The intuition is that Eq. (3.4) is calculating the difference of means, which has a connection with linear regression, and the mean value is the best least-squares point for a set of data points. Thus, noted by [136], we calculate $\phi_i(h, x) = w_j(x_j - E(x_j))$ given $h(x) = \sum_{j=1}^{M} w_j x_j + b$, which is a linear classification model such as SGDClassifier as introduced in Section 3.3.1.

During the MCBS process, the node N_{next} selection criterion is computed as follows (lines 11-13 in Algorithm 5):

$$\arg\max_{a_j} Q(N_{next}, a_j) + \lambda R(N_{next}, a_j), \tag{3.7}$$

where $Q(N_{next}, a_j)$ is the average reward score over several visits, λ is the hyperparameter that balances the exploration and exploitation. $R(N_{next}, a_j)$ is the immediate reward for choosing N_{next}, which is the $\text{SHAP}(h(\cdot), G, G_i)$ score. Finally, Algorithm 5 will output the subgraph G_{sub} with the highest risk score for cause analysis.

Table 3.1: Statistics of interaction graphs.

Type	Label	Total Graph Num.	Vulnerable Graph Num.
Homo. (IFTTT)	labeled	6,000	1,473
	unlabeled	10,000	$*$
Hetero. (5 Platforms)	labeled	12,758	3,828
	unlabeled	19,440	$*$

3.4: Evaluation and Discussion

3.4.1: Experimental Setting

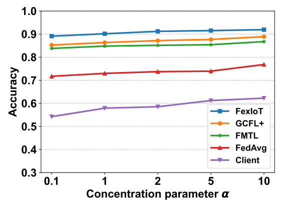
To simulate the FL training, we evaluate the dynamic clustering-based federated GNN on a high-performance computing cluster, which is equipped with Intel(R) Xeon(R) Gold 6148 2.4GHz CPUs running on CentOS 7. For collecting app description data, we use Scrapy [181] to crawl smart home app rules from the following 5 platforms:

- For SmartThings [190], we crawl rule descriptions from 185 open-source apps.
- For Home Assistant [10], we crawl rule descriptions from 574 blueprints.
- For IFTTT [98], we integrate 315,393 applets from [222] and newly crawled 1,535 applets.
- For Google Assistant [78], we crawl 480 services and 4,812 action commands.
- For Amazon Alexa [5], we crawl 2,232 services and 3,274 skill commands.

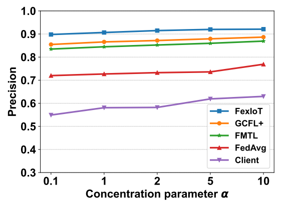
3.4.2: Clustering-based Federated Graph Learning Evaluation

FexIoT conducts fine-grained clustering-based federated contrastive graph representation learning for vulnerability detection. Note that as it is hard to build large-scale smart homes in real life, following the template of deployed smart homes, we leverage the offline graphs to produce a set of online interaction graphs for federated GNN evaluation. The only difference is that online interaction graphs reflect real-time IoT interactions.

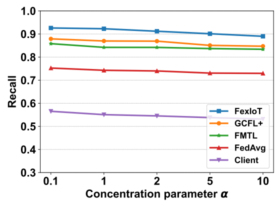
We compare FexIoT with two GNN models, four baselines, and five different Data



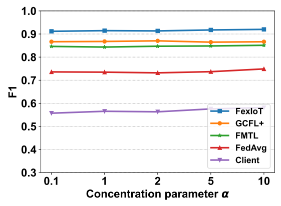
(a) GIN Accuracy



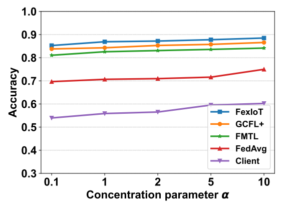
(b) GIN Precision



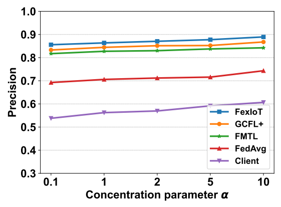
(c) GIN Recall



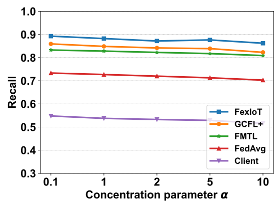
(d) GIN F1



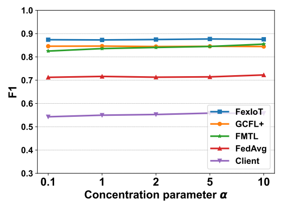
(e) GCN Accuracy



(f) GCN Precision



(g) GCN Recall



(h) GCN F1

Figure 3.2: The performance of GNNs under five different Dirichlet distribution parameters α.

distributions. The two homogeneous GNN models are: (i) GCN [108] is a convolutional network operating on graphs, and we adopt three graph convolutional layers. (ii) GIN [231] is a graph isomorphism network model, and we adopt the original model architecture. For the heterogeneous graph classification model, we choose the MAGNN model [71], which learns heterogeneous graph embeddings based on metapath aggregation.

The four federated learning frameworks are: (i) Federated multi-task learning (FMTL) [179], which groups the clients into clusters by using geometric aspects of the loss surface. (ii) Graph clustered federated learning (GCFL+) [229], which is a gradient sequence-based clustering method for graph classification. (iii) Federated averaging (FedAvg) [146], which aggregates locally-computed updates in federated learning. (iv) Self-training in clients (Client), which trains the GNN locally without any communications.

Data Distribution Configuration. We study the impacts of different data distributions on the federated GNN models. We split the IFTTT graph dataset according to Dirichlet distribution to simulate the *non-i.i.d.* dataset in each client, and we set the number of clients as 10. We draw class marginal distribution from Dirichlet distribution with the probability density function $p(x) \propto \prod_{i=1}^{k} x_i^{\alpha_i - 1}$. α is the positive concentration parameter, which we set to 0.5, 1, 2, 5, and 10. When α is close to 0, most of the generated values drawn from the Dirichlet distribution will be close to

0. Thus, we can simulate clients with unbalanced and *non-i.i.d.* datasets.

Following the above Dirichlet distribution, different clients will have different numbers of graphs. We split 80% of the IFTTT dataset as the client training dataset and 20% of the dataset as the testing dataset for each trial. During the training process, we empirically set the two thresholds ϵ_1 and ϵ_2 in Eq. (3.2), which are related to the size of model weights, to start and end the layer-wise clustering process. We found that the local clients achieve decent performance when the two values are set between 0.5 and 2. The value setting also depends on the initialization settings such as the learning rate with the corresponding optimizer. We set the learning rate as 0.001 in the Adam optimizer, ϵ_1 is 1.2 and ϵ_2 is 0.8 if they are not specified in the following experiments. We apply the weighted cross-entropy loss function to handle class imbalance for interaction graph classification. We set up the weight given to each class in the cross-entropy loss according to the inverse ratio to class frequencies. Figure 3.2 shows the average accuracy, precision, recall, and F1 value of models.

GNN Models Evaluation. We test the influence of different GNN models in the federated learning process. Note that the GNN models are used for graph representation learning in the FL process. We use SGDClassifier from Scikit-learn [164] to implement the normal or vulnerable binary classification. As shown in Figure 3.2, the GIN model achieves better performance than the GCN model, which is consistent with the previous study [231]. Moreover, we found that when the dataset is more evenly distributed (α is larger), both the GCN and GIN achieve better performance. However, the data distribution indeed has a great impact on the model performance. For example, the F1 values of the GIN model in FedAvg are 0.735 and 0.748 when α is 0.1 and 10, respectively. The GIN achieves better performance (average F1 is 0.92) when α is 10 than the performance when α is 0.1 (average F1 is 0.89). Our proposed dynamic clustering-based federated graph learning method achieves the best performance with different GNN models.

Efficacy Evaluation. Compared to baselines, our FexIoT achieves the best performance because of the design consideration of fine-grained homogeneous data features in heterogeneous data distribution. For example, for FexIoT, the accuracy is 0.891 and 0.919 when α is 0.1 and 10, respectively. For GCFL+, the accuracy is 0.852 and 0.889 when α is 0.1 and 10, respectively. The

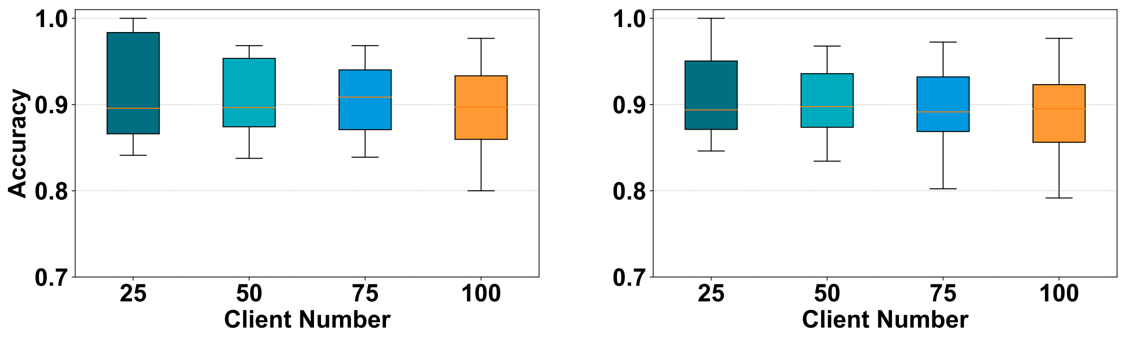


Figure 3.3: The test accuracy of client models with different numbers of clients participating in training with the IFTTT dataset (left) and heterogeneous graph dataset (right).

accuracy of FedAvg is 0.717 and 0.768 when α is 0.1 and 10, respectively. The average client accuracy is 0.542 and 0.622 when α is 0.1 and 10, respectively. FexIoT improves the accuracy by 17.4% compared with the FedAvg. FedAvg is deeply affected by the data heterogeneity, which leads to poor modeling performance with low accuracy in FL training paradigms. By contrast, FexIoT can mostly reduce the data heterogeneity effect among different datasets by training on clusters with high homogeneity. We also compare the performance of federated learning compared with centralized training. We test the GIN model in centralized training, which collects all data from clients to train a GIN model. The centralized training achieves 0.944 F1 value, while our federated GIN model achieves 0.91 average F1 value over five data distributions. Overall, our proposed FexIoT can better depict the homogeneity of the GNN models learned from different data distributions than the coarse-grained ones.

Stability Evaluation. The clustering-based methods (FexIoT, GCFL+, and FMTL) can achieve better and more stable performance than FedAvg and single client training. For example, with our proposed algorithm, the average accuracy of GIN under five different data distributions is 0.907, and the standard deviation (STD) is 0.01. For FedAvg, the average accuracy is 0.738, and the STD is 0.017. The performance of FedAvg is more divergent according to the data distribution of each client. Similarly, the performance of self-training in each client is also affected by different data distributions. Besides, our FexIoT outperforms the state-of-the-art method GCFL+, which has an average accuracy of 0.87 and STD is 0.012. Facing the heterogeneity in datasets, the GCFL+

considers the similarity of parameters of a whole model, which can be fluctuating due to the changes in parameters of different layers of client models. By contrast, our FexIoT implements clustering based on fine-grained bottom-up layer-wise parameters, which is less affected by fluctuating layers. Overall, our FexIoT is more stable under different data distributions than other FL methods.

Scalability Evaluation. We simulate different numbers of clients (25, 50, 75, 100) joining the FexIoT on two datasets. We use GIN on the IFTTT dataset and MAGNN on the heterogeneous graph dataset if not mentioned. The concentration parameter α of the Dirichlet distribution is 1. Figure 3.3 is a box plot, in which minimum, maximum, median, first quartile, and third quartile accuracy are reported. On the IFTTT dataset, the third quartile accuracies of 25, 50, 75, 100 clients are 0.869, 0.879, 0.882, and 0.873, respectively. Therefore, even though the number of clients increases, 75% of clients can achieve more than 86% accuracy, which shows the high scalability of FexIoT. When the number of clients is larger (e.g., 100), the results show more divergence as the minimum accuracy is 0.8, and the maximum accuracy is 0.977. When the number of clients increases, due to the fixed size of the entire dataset, the size of the dataset for each client decreases, which affects the evaluation results. Nevertheless, the results show that FexIoT can effectively aggregate clients that have similar data distribution to boost the model performance.

Drifting Interaction Pattern Evaluation. The methods that rely on known patterns will become fruitless when there are drifting samples. Therefore, we need to identify the possible drifting graph samples with unknown interaction patterns. We evaluate drifting interaction pattern detection on the unlabeled dataset, to check if the randomly generated graph dataset contains drifting samples. We show the clustering results in Figure 3.4 on 1500 randomly sampled graph feature representations learning with federated contrastive graph learning. The data is processed with TSNE dimension reduction. The centroid of each class is the white cross, and the possible drifting samples are in the red circle. The six types of known interaction vulnerabilities are clustered in gray dotted boxes. Through our federated contrastive learning, the model can learn well the features of six types of vulnerability patterns and the normal graph pattern, which are separable in hidden space shown in Figure 3.4. Based on the drifting interaction pattern analysis in Section 3.3.2,

Table 3.2: Comparison of different systems with testbed data.

Method	Accuracy	Precision	Recall	F1
HAWatcher [70]	0.82	0.83	0.87	0.85
DeepLog [59]	0.74	0.78	0.79	0.78
IsolationForest [131]	0.63	0.74	0.61	0.67
FexIoT	0.9	0.9	0.93	0.91

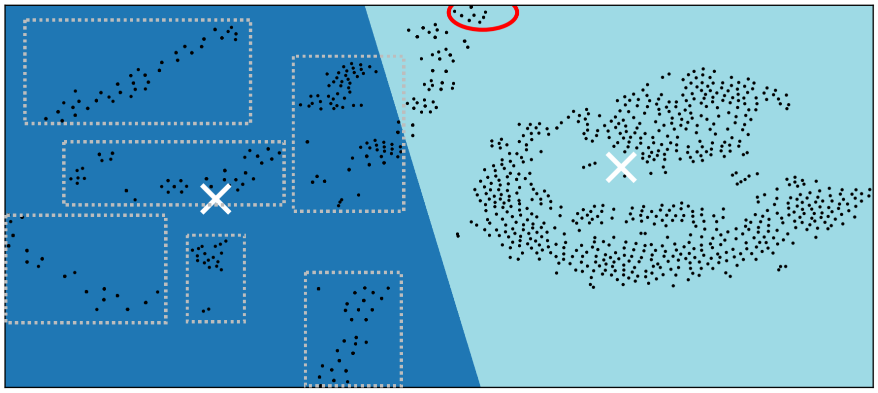


Figure 3.4: K-means clustering on 1500 randomly sampled graph representations with TSNE dimension reduction.

we found 63 and 104 potential drifting samples in the IFTTT dataset and heterogeneous graph dataset, respectively. After manually checking, we found three new types of vulnerability patterns: (i) Automation action is reverted over time. (ii) Another action can generate fake automation conditions. (iii) Non-automation settings can block the existing actions of smart devices. In this way, FexIoT can reduce the false-alarming rate while the security rule-based querying methods in the traditional database cannot discover new patterns and even incur high false-positive or false-negative rates because of the drifting samples.

System Comparison. We compare FexIoT that is trained across 50 clients with existing vulnerability detection systems: (i) HAWatcher [70] is a search-based system that extracts secure rule interaction templates and evaluates device interaction. We use the extracted templates to discover vulnerabilities in the test dataset. (ii) DeepLog [59] models event logs as a language sequence and uses Long Short-Term Memory (LSTM) to learn log patterns. We train LSTM on normal logs and find anomalies that deviate from the training data. (iii) IsolationForest [131]

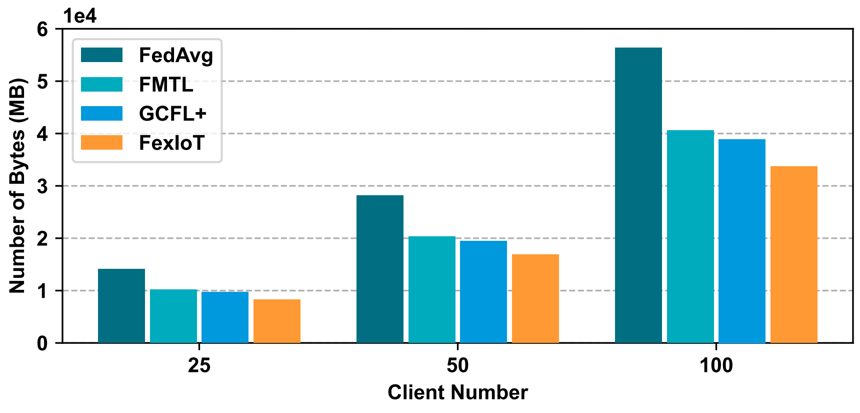


Figure 3.5: Communication cost with different number of clients.

is a density-based anomaly detection method with a tree structure, which is implemented by Scikit-learn [164], and the input is a data vector that includes device status. We use 600 online interaction graphs in Section 3.4.1 that are integrated with event logs and descriptions as the test dataset. The system performance is shown in Table 3.2. From the results, we find that all methods fail to differentiate between normal user interruptions (e.g. manually turning on a light) and malicious attacks, which cause false positives. Feedback from users should be provided to update such correlations and reduce false alarms. FexIoT outperforms the existing methods due to the following reasons: (i) HAWatcher only extracts binary rule templates, which can hardly cover long-term complex correlations such as the correlation between an air conditioner and temperature events, and (ii) DeepLog and IsolationForest cannot effectively mine interaction correlations across different events.

Communication Cost Evaluation. We measure the communication cost during the training process by computing the total data transferred (download and upload) between the server and clients. As shown in Figure 3.5, the total transferred data is less than 40 GB in 60 rounds with 100 clients, which is acceptable for wired network bandwidth. FexIoT saves 40.2% communication overhead compared with FedAvg [146]. FedAvg needs to aggregate the whole model during the FL process, and FMTL [179] and GCFL+ [229] also share the whole model but within different clusters. The low cost of FexIoT is attributed to our proposed layer-wise clustering-based FL method. At the initial stage, only the parameters of the first layer are uploaded to a server for clustering. Then, based on the previous clustering result, the upper layer parameters of models are uploaded to the server, and the server sends parameters of different layers back to the corresponding client clusters. Clients in the same cluster share more layers than clients that are in different clusters. Thus, it reduces the number of parameters transmitted between the server and clients.

3.4.3: Vulnerable Interaction Cause Analysis Evaluation

After vulnerability detection, we implement qualitative and quantitative evaluations to show the efficacy and efficiency of the proposed vulnerable interaction cause explanation method. We randomly select 100 interaction graphs that contain vulnerable interactions, which are reported by the GCN model on the unlabeled IFTTT dataset. Among the 100 graphs, we identify four graphs that are benign after manually checking, which are false-positive results from the GCN model.

We compare our method FexIoT with two baseline methods: SubgraphX [241] and MCTS_GNN. SubgraphX is a general graph explanation method that applies Monte Carlo tree search with shapely value to explore various subgraphs, while MCTS_GNN applies Monte Carlo tree search with the prediction score of the GNN model to explore subgraphs. Other graph explanation methods such as PGExplainer [137] and GNNExplainer [238] can only identify discrete edges or nodes, which cannot be used for the interaction vulnerability explanation among nodes. The SubgraphX and MCTS_GNN are implemented by the DIG [132] library. We apply GCN as the vulnerability detection model because the proposed vulnerable interaction cause analysis method does not rely on specific vulnerability detection models.

Example Illustration. To visually demonstrate the superiority of our explanation method, we give two intuitive examples in Figure 3.6. The first row illustrates the example of the false positive of the GCN model prediction, while the second row illustrates the correct prediction. For the first row in Figure 3.6, the interaction graph is predicted to contain vulnerable interactions, but it is deemed benign after manual inspection. The interaction graph describes that when the door opens, the water flow will run with the switch turned on with a notification sent to users. The user can notify the app to turn on the camera. Then, if the smoke is detected, the door unlocks and the

Table 3.3: The content of indexes in Figure 3.6.

Index	Rule
23	Connect to the WiFi
47	Turn camera off
62	If the fan runs, turn on water flow switch
174	Turn on cameras if lights are off
281	If smoke is detected, the fan starts
1076	Turn on air conditioner if plug is on
1177	Turn the camera on if get notified
1215	Tap to turn on camera off
1291	Turn on plugs if door unlocked
2184	If smoke is detected, unlock the door and start fan

exhaust ventilation fan starts. This interaction graph does not contain interaction vulnerabilities, as shown in Figure 3.6, our FexIoT offers a minor misleading explanation subgraph, as it only identifies the concise subgraph to explain the prediction. By contrast, other methods try to find a larger subgraph to explain the prediction result, which causes more confusion for the inspector.

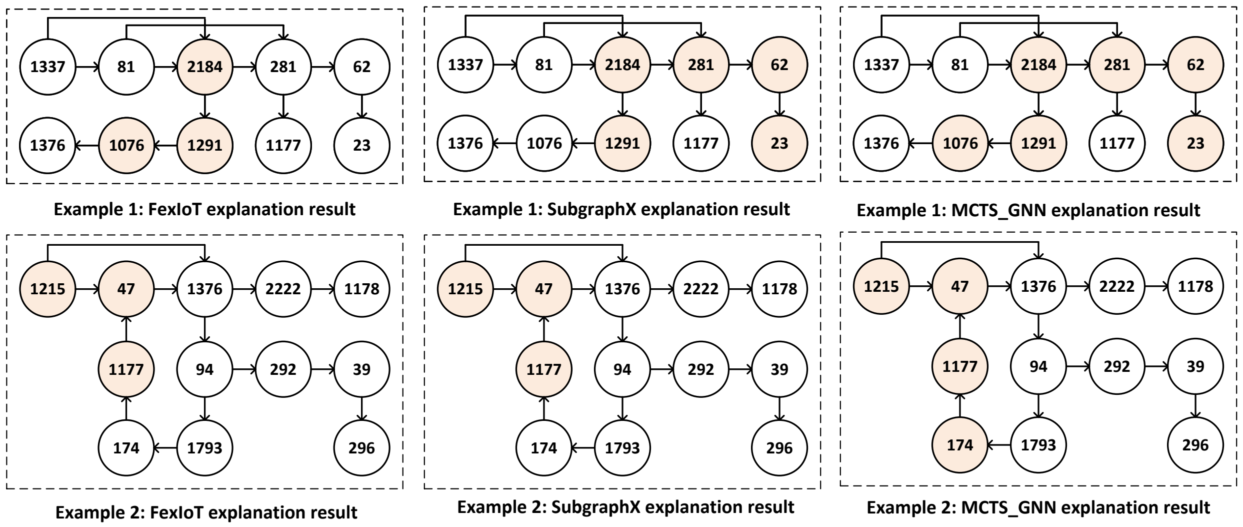

1337
81
2184
281
62
1376
1076
1291
1177
23
Example 1: FexIoT explanation result
Example 1: SubgraphX explanation result
Example 1: MCTS_GNN explanation result
1215
47
1376
2222
1178
1177
94
292
39
174
1793
296
Example 2: FexIoT explanation result
Example 2: SubgraphX explanation result
Example 2: MCTS_GNN explanation result

Figure 3.6: Vulnerability explanation comparison.

For the second example in Figure 3.6, the GCN model prediction is correct. It describes that when the user taps to turn off the camera, the camera will be turned off and the action will be recorded in a Google spreadsheet. When a new spreadsheet subscriber is added, it will send a

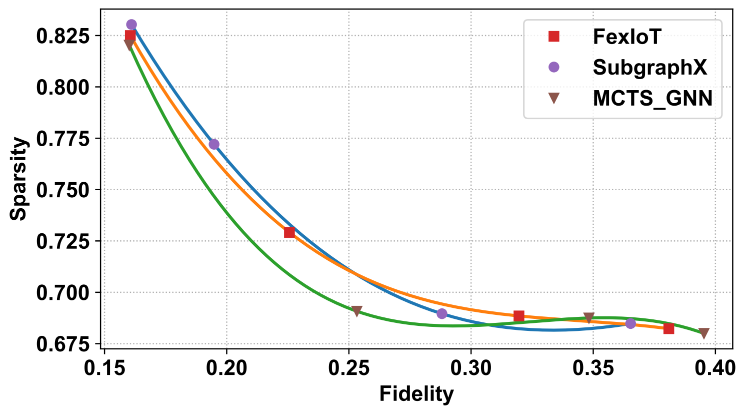


Figure 3.7: The comparison of Fidelity and Sparsity scores.

notification to turn on the camera. FexIoT, SubgraphX, and MCTS_GNN all can recognize the key subgraph that causes the interaction vulnerability that turns off the camera within a loop. For both cases, FexIoT can provide a reasonable vulnerability explanation with better visual explanations.

Fidelity and Sparsity. Besides the visualization to evaluate the vulnerability explanation methods, we use the *Fidelity* and *Sparsity* [168] to quantitatively measure the explanation results. The Fidelity metric refers to the difference between two prediction scores before and after removing a part structure of a graph. The Sparsity metric refers to the percentage of the structure that is recognized as important by the explanation model. There is a trade-off between Fidelity and Sparsity. A high Sparsity score means a smaller structure, which tends to be low Fidelity (less important). For real-life cases that lack the ground truth, the Fidelity and Sparsity scores can help measure the explanation quality.

We test the Fidelity and Sparsity scores of 50 randomly picked interaction graphs. The scores depend on the structure of interaction graphs, which fluctuate among different cases. Half of the tested cases have relatively high Fidelity which is greater than 0.3, and low Sparsity which is smaller than 0.7. For better visualization, we plot the curve of Sparsity concerning Fidelity for four cases as shown in Figure 3.7. The four cases have higher Fidelity and lower Sparsity, or lower Fidelity and higher Sparsity. Our method FexIoT can strike a decent balance between Sparsity and Fidelity scores, which means FexIoT can accurately identify the concise subgraphs that are the possible causes of the vulnerabilities. The results indicate that our explanation method can find the important

Table 3.4: Runtime efficiency with different datasets.

	Graph Construction Time (s)	Prediction Time (s)	Vulnerability Analysis Time (s)	Model Size (MB)
IFTTT	17.19	0.52	2.18	5.48
Hetero.	976.99	0.61	3.64	6.13

yet concise subgraph for prediction explanation, which is more faithful for interaction analysis.

Vulnerability Explanation Efficiency. As shown in Table 3.4, the 12,758 heterogeneous graphs generation takes 980s. The size of the MAGNN model for heterogeneous graphs is only 6.13MB, which makes it feasible to train the model on devices such as a Raspberry Pi. The prediction time is 0.61s on average, which is related to the graph size. For vulnerability explanation, the average cause discovery time is 3.64s for an interaction graph. It takes 182s for 50 graphs with 18 nodes on average. Note that the time is related to the algorithm parameters such as the number of MCTS iterations, and the kernel SHAP samples. **Overall**, the FexIoT achieves vulnerability analysis with high efficiency.

3.4.4: Discussion

Security and Privacy

We acknowledge that FexIoT itself faces security and privacy issues. For example, it might suffer from Sybil attacks and data poisoning attacks [56, 72, 103, 256] when an attacker controls multiple clients to attack the system. One possible solution is to enhance the software and network security (e.g., using firewalls, CAPTCHAs, or device-specific asymmetric keys) in smart homes to protect the internal FL model. Moreover, there are several defense methods that can identify sybils based on the diversity of client updates [72], training loss from randomly selected sets of clients [103], and feature importance [256], which could be integrated into FexIoT. To enhance the data privacy, we will add differential privacy [61, 134, 226] and secure aggregation mechanisms [19, 42, 129] to FexIoT in the future.

Data Labeling

Providing ground truths is necessary for scientific evaluation. Existing works [57, 122, 242] have proposed methods for efficient data annotation. In this chapter, we detailed how to manually label the dataset with cross-validation. In the future, we plan to combine experts, amateurs, and ML models for cross-validation to further enhance the quality of labels for IoT data. For example, experts can label a few samples and develop few-shot learning-based models. ML models and amateurs can collaboratively label new data. Iteratively, the newly labeled data could be used to re-train the model.

3.5: Summary

IoT data security and privacy management are essential for many IoT applications. In this chapter, we investigate how IoT rule configuration data could expose interaction vulnerabilities across heterogeneous closed-source platforms. We present and implement a novel IoT data management solution, FexIoT, that leverages federated and explicable graph learning to analyze large-scale IoT interaction data. We design the fine-grained dynamic clustering-based federated contrastive graph representation learning algorithm to discover interaction vulnerabilities and tackle the concept drift problem of smart home data. We propose the MCBS-based search method with SHAP value to search and measure the risk of each subgraph. With the data collected from 5 real-world IoT platforms, we demonstrated the superior performance of FexIoT in detecting and tracking the root causes of interaction vulnerabilities.

CHAPTER 4: ACOUSTIC-BASED FACIAL EXPRESSION RECOGNITION[3]

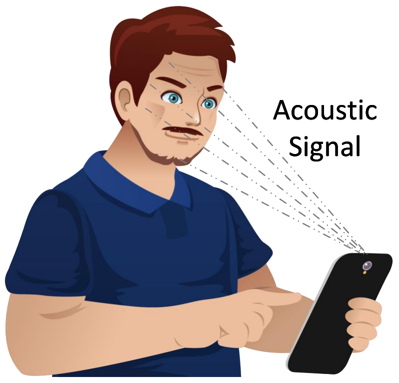
4.1: Introduction

With the emergence of new media in the digital era, a variety of social media services are craving users' attention. Fine-grained emotional reaction understanding is pivotal to facilitating a user's interaction with digital content. Traditionally, crowd-sourced ratings and reviews have been used for evaluating users' feedback on services. However, they are too coarse-grained to provide real-time spontaneous feedback. To provide a more personalized service, we need an accurate and robust approach to recognizing the users' emotions and acquiring spontaneous feedback.

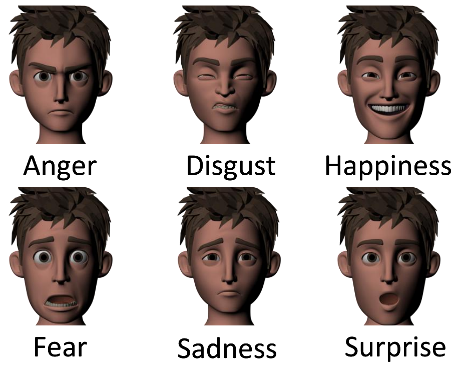
A number of techniques have been proposed to recognize emotions, expressed by various biometric features, such as facial features [32, 75, 155], speech features [156], or heartbeats [250]. Nevertheless, as a universal form of nonverbal communication, facial expression is recognized as the most direct way of understanding human emotions [126]. There are six widely adopted facial expressions (FEs): anger, disgust, fear, happiness, sadness, and surprise [63]. These FEs can be modeled by the Facial Action Coding System (FACS), which includes action units (fundamental action muscles) and action descriptors (unitary movements of several muscle groups) [41]. When people make different facial expressions, different facial muscles will move, which could be captured by various sensing signals.

Existing facial expression recognition (FER) methods can be categorized as camera-based [23, 163, 183], radio-based [36, 40, 81] and acoustic-based [75] expression recognition.

[3]This chapter is based on previously published work by Guangjing Wang, Qiben Yan, Shane Patrarungrong, Juexing Wang, and Huacheng Zeng titled "FacER: Contrastive Attention based Expression Recognition via Smartphone Earpiece Speaker" published in the Proceedings of IEEE Conference on Computer Communications (INFOCOM). DOI: 10.1109/INFOCOM53939.2023.10228985 [216]. © 2023 IEEE. Reprinted, with permission, from Guangjing Wang, Qiben Yan, Shane Patrarungrong, Juexing Wang, and Huacheng Zeng, "FacER: Contrastive Attention based Expression Recognition via Smartphone Earpiece Speaker" (paper and IEEE titles are the same), August 2023.



(a) FacER application scenario.



(b) Six facial expressions [7].

Figure 4.1: Facial expression recognition using a smartphone.

However, the existing camera-based methods raise privacy concerns due to their continuous video recording on a device. For instance, FaceWarehouse [23] collects RGBD images of different expressions from different users. Yet, serious privacy concerns impede the wide adoption of such methods in a real-world scenario. Moreover, these camera-based methods cannot accurately recognize facial expressions when the users cover their faces with masks. Other alternative methods require extra sensing devices, affecting the usability. For example, WiFace [36] uses a WiFi router with three antennas placed at a specific position for FER, but it requires additional hardware and configurations. Similarly, PPGface [40] requires expensive wearable devices with photoplethysmography sensors.

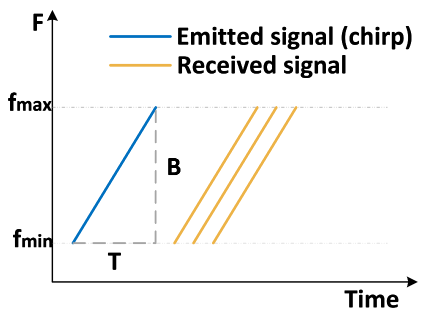
We propose FacER, a **Fac**ial **E**xpression **R**ecognition system based on near-ultrasound acoustic sensing on a smartphone. We utilize commodity smartphones to emit near-ultrasound signals (19-23 kHz) towards the user's face. As shown in Figure 4.1a, the microphone on the smartphone will receive the reflected echoes from the face surface, which will carry the facial expression information. By examining the fine-grained echo patterns, FacER can differentiate six different types of facial emotional expressions as shown in Figure 4.1b. We demonstrate that in a scenario where a user holds a smartphone at a relatively stable distance (*e.g.* 20-50 cm), FacER can recognize six universal facial expressions with more than 85% accuracy. Compared with the state-of-the-art acoustic-based SonicFace [75], FacER uses a commodity smartphone without requiring a customized microphone array, providing a more convenient and usable solution.

There are two main challenges in designing FacER. First, acoustic interference, such as the multipath of reflected signals and environmental noises, can significantly impact recognition performance. Apart from the echoes reflected from the face, the microphones can also receive echoes from the surrounding obstacles. It is imperative to mitigate or remove any undesired signals. However, even with the application of various noise-reduction techniques, the environmental noises at a similar frequency to the emitted signal could persist. There could be a similar distance between the smartphone and obstacles as the distance between the smartphone and the user. Besides, there could exist low-frequency and high-frequency environmental noises. Therefore, we propose a contrastive external attention-based learning model to depict more robust facial expression feature representations from the noisy data. In this way, the model can distill the universal features of expressions while eliminating background noises.

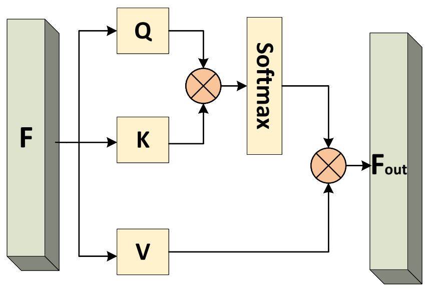
Second, different users express facial expressions in different manners, and even the same user could express them differently at different times. This will result in the domain adaptation issue, a common issue in machine learning (ML) models, where a model trained on a labeled dataset (source domain) has limited performance on a testing dataset (target domain). This is because the target domain dataset has a distribution shift with the source domain dataset, which violates the independent and identically distributed (*i.i.d.*) assumption of ML models. Therefore, we propose a domain adaptation contrastive learning algorithm to align the distribution of the source domain and target domain dataset. In this way, FacER can achieve consistent performance in recognizing various expressions across different users.

We evaluate FacER on a dataset collected from 20 volunteers of different ages, genders, and skin colors in various environments. We show that FacER can effectively recognize six different facial expressions from different users. Remarkably, it achieves more than 90% test accuracy when the training and testing datasets have the same distribution. It achieves more than 85% accuracy when training and testing on different sets of users. In summary, we make the following contributions:

- We design FacER, which uses a contrastive external attention-based acoustic facial



(a) Illustration of FMCW.



(b) Illustration of self-attention.

Figure 4.2: Illustration of Preliminaries.

expression recognition model to learn representative and robust facial expression features while eliminating the background noise.

- We propose the domain adaptation contrastive learning algorithm to align the training and testing data distributions, which can largely mitigate the negative effects of variations in users' facial expressions.
- We implement the smartphone-based system FacER and perform the evaluations in various real-life scenarios. FacER outperforms the state-of-the-art approaches by more than 10% in recognition accuracy with high mobility and convenience.

4.2: Background

In this section, we introduce the background of acoustic signals, the preliminary knowledge of attention mechanisms and contrastive learning.

4.2.1: Acoustic Signal

The acoustic signal refers to a coded chirp signal transmitted by a device. Especially, the chirp sound signal is the frequency-swept signal, which is modulated in frequency as shown in Figure 4.2a. Particularly, the chirp signal can be regarded as the component of sawtooth modulation in the Frequency-Modulated Continuous Wave (FMCW), which changes its operating frequency during the measurement. In FMCW, the frequency of the signal will periodically increase or decrease during transmission. The differences in frequency between the transmitted and received signal are proportional to the time delay Δt. Therefore, the FMCW can measure the small movement of the target, which is calculated as follows:

$$R = \frac{v_0|\Delta t|}{2} = \frac{v_0|\Delta f|T}{2B}, \tag{4.1}$$

where R is the distance between the sound source and the reflecting object, v_0 is the speed of sound (340 m/s) at 20 °C, Δt is the delay time, and Δf is the measured frequency difference. B is the chirp frequency bandwidth, and T is the chirp periodic time. The duration of the transmitted waveform T should be greater than the required receiving time for the distance measuring range. We use $\frac{B}{T}$ to measure the frequency shift per unit of time. Therefore, with the features of the FMCW, the chirp signal can help group reflections from various distances into multiple range bins.

4.2.2: Attention Mechanism

Similar to the human visual system, attention mechanisms [207] aim to focus limited attention on key information, which saves resources and distills the most essential information. The basic idea of attention mechanisms is to combine all of the encoded input features in a weighted fashion, with the most important features receiving the highest weights. The self-attention mechanism improves the representation at each position by combining features from other positions in a sample (*e.g.* image), and it captures long-range dependencies.

Given a feature map $F \in \mathcal{R}^{N \times d}$, where N is the number of elements and d is the feature dimension of each element, by multiplying three different random initialized weight matrixes, self-attention projects the F into a query matrix $Q \in \mathcal{R}^{N \times d}$, a key matrix $K \in \mathcal{R}^{N \times d}$, and a value matrix $V \in \mathcal{R}^{N \times d}$. The self-attention is represented as follows:

$$F_{out} = softmax(QK^T)V, \tag{4.2}$$

where $softmax(QK^T)$ is the attention matrix, and the F_{out} is the improved feature representation of the input F.

4.2.3: Residual Attention Network

Residual attention network (RAN) [213] is a type of convolutional neural network. The RAN contains multiple attention modules that generate attention-aware features. In each attention module, there are trunk branch $T(x)$ and mask branch $M(x)$. The $T(x)$ performs feature processing, and $M(x)$ softly weight output features of $T(x)$. The output of the attention module is

$$H_{i,c}(x) = M_{i,c}(x) * T_{i,c}(x), \tag{4.3}$$

where i covers all spatial positions, and c is the index of a channel. However, directly stacking multiple attention modules would degrade the model performance. Therefore, the residual learning mechanism is applied to optimize the performance. The main design of residual learning is the skip connections that jump over some neural network layers, containing nonlinearity activations and batch normalization. Thus, the modified output H of the attention module is

$$H_{i,c}(x) = (1 + M_{i,c}(x)) * F_{i,c}(x), \tag{4.4}$$

where $F_{i,c}(x)$ is the original features, which can be approximated when M(x) is 0. This method is called attention residual learning. The facial expression echos contain multiple facial muscle movements, as well as the background noise. Therefore, it is essential for the neural network to capture different important aspects of expressions.

4.2.4: Contrastive Learning

A domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ includes the feature space $\mathcal{X}$ and marginal probability distribution $P(X)$. If two domains are different, they have different $\mathcal{X}$ or $P(X)$, but the label space is the same. In our context, the source domain refers to the dataset with labels while the target domain indicates a dataset with different distributions.

Contrastive representation learning aims to learn an embedding space where dissimilar samples are spread out and similar samples remain close together. Normally, a positive pair refers to a pair of samples that have the same label, and a negative sample pair has different labels. The contrastive loss with two samples is defined as

$$\begin{aligned} \mathcal{L}_{\mathrm{c}}(\mathbf{x}_i, \mathbf{x}_j, \theta) = \mathbb{1}[y_i = y_j] \| f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j) \|_2^2 \\ + \mathbb{1}[y_i \neq y_j] \max(0, \epsilon - \| f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j) \|_2)^2, \end{aligned} \tag{4.5}$$

where $\mathbb{1}[y_i = y_j]$ means the value is 1 if two samples have the same label, otherwise is 0. $\mathbb{1}[y_i \neq y_j]$ means the value is 1 if two samples have different labels, otherwise is 0. ϵ is the upper bound distance to avoid the exceptional contribution of some dissimilar pairs.

The supervised contrastive loss [106] is defined as follows when the training objective includes multiple positive and negative pairs in one batch:

$$\mathcal{L}_{\mathrm{c}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} log \frac{exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} exp(z_i \cdot z_a / \tau)}, \tag{4.6}$$

where I is a set of samples x within a batch, $A(i) \equiv I \backslash \{i\}$, $P(i)$ is a set of indices of all positives in the multiviewed batch, $|P(i)|$ is the cardinality, $z = Proj(Enc(x))$ is the encoded feature representation by an encoder model $Enc(\cdot)$ and a projection layer $Proj(\cdot)$ such as a linear layer. The $\cdot$ denotes the inner product, and τ is a temperature parameter to adjust the final results.

4.3: System Design

In this section, we design the acoustic sensing and signal pre-processing techniques to prepare the sensing data. Then, we propose the contrastive external attention-based domain adaptation model for acoustic facial expression recognition.

4.3.1: Acoustic Sensing Design

A unique facial expression contour is a distinct collection of various reflecting surfaces. As different objects absorb and attenuate sound waves to different degrees, it is possible to differentiate between the reflected echoes from objects and those from facial expressions [254].

Signal Generator

On a smartphone, there are usually one main speaker and microphone at the bottom or on the back, and an earpiece speaker and microphone at the top of the phone body. Considering that the earpiece speaker has a proper position to illuminate a user's face as shown in Figure 4.1a, we select the earpiece speaker for emitting the acoustic signal. Similarly, considering the gesture of holding a phone, the top microphone is chosen since it is less affected by the hand.

The acoustic signal should satisfy the following properties. (i) The period of the signal should be moderate to minimize the overlap of echoes from various distances. (ii) The signal should be distinguishable from the background noise in the frequency domain, while the noise frequency is mostly under 8 kHz. (iii) The signal ought to be inaudible in real-world scenarios. Therefore, considering that the change of facial expression happens within 1 second, we choose a chirp signal of 25 milliseconds with frequency sweeping from 19-23 kHz to compose the inaudible acoustic signal with fading at the start and the end. In this way, we can better capture the expression features caused by muscle movements and filter out echoes from different obstacles. According to the Nyquist sampling theorem, the sampling rate is set as 48 kHz. FacER leverages the earpiece speaker to periodically emit the near-ultrasound signals and simultaneously uses the microphone to receive the signals reflected by the face. We set up the time interval as 50 milliseconds to allow all the reflected signals from the previous chirp to be received, such that we can separate two chirps before the following chirp is transmitted.

Noise Removal

Expression-irrelevant signals from various background noises, nearby people and obstacles should be filtered out. To achieve that, we first apply a 19-23 kHz band-pass filter to keep the desired frequency band and filter out the background noise. There remain three main types of signals in the received recording: (i) the *direct path signal* directly travels from the speaker to the microphone; (ii) the *major echo signal* is the mix of echoes from the facial contour, which is the interest of FacER; and (iii) the *noisy echo signals* are echoes from different obstacles in the environment because of

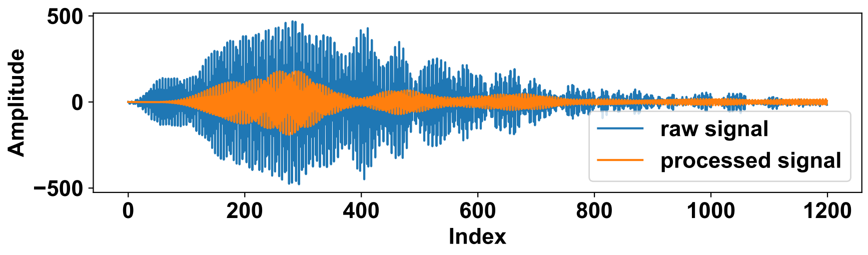


Figure 4.3: The raw signal and the signal after noise removal.

the multipath of reflected signals. Figure 4.3 shows a segment of the raw signal (in blue) and the processed signal (in orange) after filtering out the background noise and the direct path signal noise.

To remove the interference of the *direct path signal*, inspired by AIM [144], we use a separate speaker and microphone to record the direct transmission by positioning the devices in a clean and quiet space. Therefore, we remove the direct path signal from the received samples by minimizing $||S - cS_d||$, where S denotes the received samples, S_d denotes the pre-recorded direct signals, and c is a scaling coefficient to achieve optimal cancellation which is set as 0.9 in our experiments. In this way, we remove the *direct path signal* from the speaker to the microphone.

Then, we proceed to remove the *noisy echo signals*. In the FacER application scenario, a user is facing a phone, so we assume that there is a relatively stable and static distance between the phone and the user's face. We consider filtering out the *noisy echo signals* from the nearby obstacles at different distances. The FMCW provides distance measurement, which is an important tool for differentiating among different echoes when more than one source of reflection is received. A comfortable distance between human eyes and the phone is $25 - 50\ cm$ [254]. Therefore, based on Eq. (4.1), we can calculate the desired frequency shift as:

$$|\Delta f| = \frac{2RB}{Tv_0}. \tag{4.7}$$

Thus, $|\Delta f|$ is between 235 Hz and 470 Hz. We further analyze the FMCW distance measurement resolution. Given the minimum measurable frequency shift $\Delta f_{min} = 1/T$, we can compute the

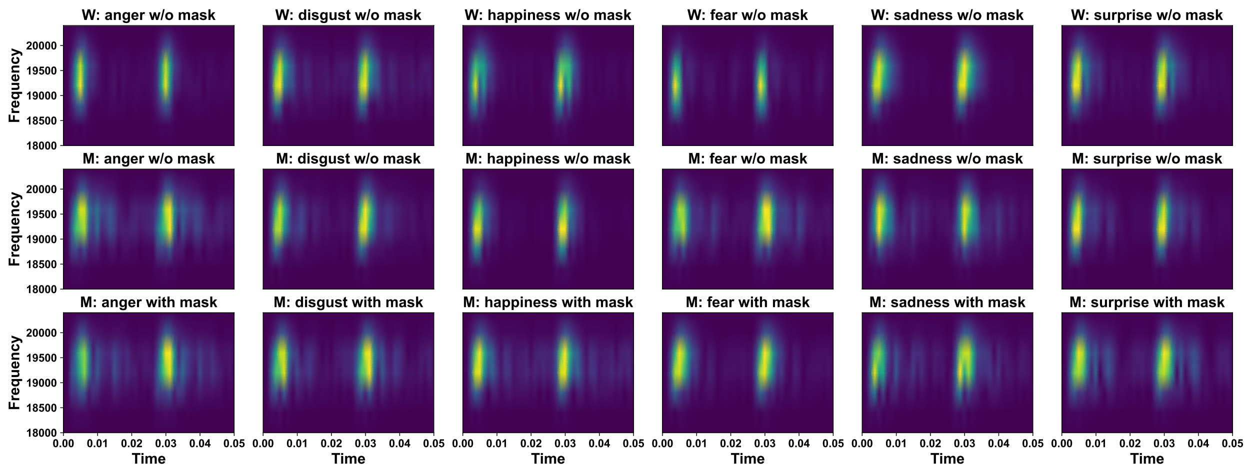


Figure 4.4: The spectrogram of six expressions in 50 milliseconds. The first row is from a woman without a mask, the second row is from a man without a mask, and the third row is from the same man with a mask.

resolution d_r that FMCW separates mixed echoes as:

$$d_r = \frac{v_0 \Delta f_{min}.T}{2B} = \frac{v_0}{2B}. \tag{4.8}$$

Thus, d_r is $\frac{340m/s}{2\times 4000s^{-1}} = 4.25\ cm$. The resolution of the *major echo signal* corresponding to a single sample is $\frac{v_0}{2F_s} = 3.54\ mm$, where F_s is the sampling frequency 48 kHz.

4.3.2: Contrastive Attention-based Domain Adaptation

We use the Short-Time Fourier Transform (STFT) with the Hann window to process the signal, which outputs the complex amplitude, and we compute the absolute values of the STFT values. The generated spectrogram is used as input for our proposed model in Algorithm 6. Figure 4.4 shows the spectrogram of the segmented major echo signals of different expressions from two volunteers. We can observe that, for the same person, different expressions will yield different spectrograms. For the same expression, different people have different ways of making expressions.

However, we make two observations that would affect the modeling performance. First, it is nearly impossible to entirely remove noisy echo signals from different obstacles at various distances in some scenarios. We set up the desired frequency shift $|\Delta f|$ as 500 Hz in Figure 4.4. However,

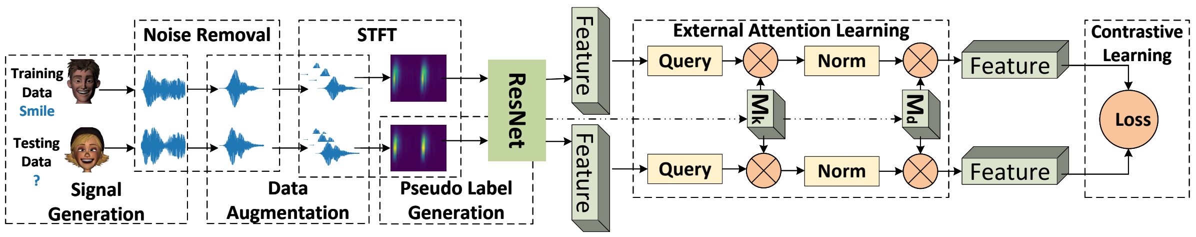


Figure 4.5: The contrastive attention-based domain adaptation model for acoustic facial expression recognition using smartphone earpiece speaker.

there could be minimal multipath changes caused by body motions or objects between the face and the phone, which is hard to filter out since the resolution d_r for FMCW to separate mixed echoes is 4.25 cm. For example, the second row in Figure 4.4 is from a man without a mask, while the third row is from the same man with a mask. We can observe subtle differences between the corresponding spectrograms (e.g., “surprise”) in the two rows. Considering that the image-based expression recognition model can cope with various backgrounds around the face in an image to extract the facial expression-related features. Similarly, we aim to design the model to pay attention to the acoustic facial expression features.

Second, normal ML models face the poor generalization problem when there is a distribution shift between the training and testing datasets [258]. For example, as shown in the first and second rows of Figure 4.4, for the same expression, different people will have different ways of expressing facial emotions, as manifested in the differences in the corresponding spectrograms. These differences will cause distribution shifts. Therefore, we should align the training and testing data distribution before training a classification model. Moreover, the model should only require a limited amount of enrollment data to improve its usability. Yet, it is challenging to extract useful features that can identify expressions from weak signals with limited acoustic samples.

To tackle these challenges, we have developed a contrastive attention-based domain adaptation model aimed at extracting consistent acoustic facial expression features, as depicted in Figure 4.5. The core principle of domain adaptation in this context is to minimize the disparities in feature representations of expressions across different domains. Yet, acquiring a sufficiently diverse and extensive dataset to train deep learning models for robust facial expression feature extraction poses a significant challenge. To overcome this, our initial step involves augmenting the dataset to ensure a broader representation of facial expressions from various populations.

Data Augmentation

Data augmentation, which involves adding modified versions of existing data or generating new synthetic data from existing data, is a typical technique to address the data deficiency problem. First, it can mitigate the model overfitting problem when the original dataset is relatively small. Second, contrastive learning learns discriminative representations by bringing together positive pairs and separating negative pairs. The different augmented views of each sample can compose positive pairs that have the same labels, which enhances the model's discriminative ability.

We propose to use two methods for acoustic expression data augmentation. First, we shift the acoustic expression signal segment by the same distance. In accordance with the inverse square law of sound propagation, the amplitude of the signal is changed by a scale equal to the inverse square of the distance (*e.g.*, 0.3 meters). Second, for the generated spectrograms, we produce different versions of the spectrogram by multiplying the magnitudes of the spectrogram by a scalar. We generate the scalar by sampling from a Gaussian distribution with the mean being 0 and the standard deviation being 0.1. Considering our scenario when a user is holding a smartphone, a small device rotation at a fixed position creates negligible changes in the signal due to the omnidirectional nature of speakers and microphones, therefore, we only consider the changes in the device positions for acoustic signal transformation. In this way, we can grow and enrich our acoustic facial expression dataset for contrastive external attention learning.

Pseudo Label Generation

A domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ includes the feature space $\mathcal{X}$ and marginal probability distribution $P(X)$. If two domains are different, they have different $\mathcal{X}$ or $P(X)$, but the label space is the same. Suppose we have a source domain with a fully-labeled expression training dataset D_s, and an unlabeled testing dataset D_t is the target domain, which has the same categories as the source domain. The first problem is how to form the positive pairs from the same category of D_s and D_t

when the labels in D_t are unknown.

Inspired by DeepCluster [24], we generate pseudo labels for unlabeled data in D_t according to the highest category probability. Specifically, we propose to use K-means clustering to generate pseudo labels, re-train the current model and adjust the noisy pseudo labels iteratively. For each iteration, we first calculate the centroid for each class in the target domain:

$$c_k^{(i)} = \frac{\sum_{x_t \in \mathcal{X}_t} \delta(z_t) g_t(x_t)}{\sum_{x_t \in \mathcal{X}_t} \delta(z_t)}, \tag{4.9}$$

where $c_k^{(i)}$ is the centroid for class k at the i_{th} iteration, $\delta(z) = \frac{exp(z)}{\sum_m exp(z_m)}$ is the softmax function that is used to output category probability. $g_t(x_t)$ outputs the representation of an acoustic expression sample x_t, and $z_t = h(g_t(x_t))$ is the linear transformed representation of x_t. The centroids can characterize the distribution of categories within the target domain. The pseudo labels are obtained via the nearest centroid:

$$y_t = \arg\min_k cos(g_t(x_t), c_k), \tag{4.10}$$

where $cos(\cdot)$ is the cosine distance between the representation embedding $g_t(x_t)$ and the centroid embedding c_k. As a result, we can obtain pseudo labels for the target domain dataset and compose the positive and negative pairs using both source and target domain datasets. Then, we iteratively update the model parameters by minimizing the loss function in Eq. (4.13) described below.

Attention-based Expression Learning

Given the challenge of eliminating all noisy echoes, the model needs to learn to differentiate between echoes related to facial expressions and those stemming from background noise. While self-attention mechanisms are commonly employed to learn robust features, their quadratic complexity and lack of consideration for potential correlations between different samples limit their effectiveness. Each facial expression generates multiple data samples (at a rate of 0.1 seconds per sample for an expression lasting approximately 1 second). By identifying and leveraging the correlations among these samples, the model can be trained to concentrate on the consistent and common features specific to acoustic facial expressions, enhancing its accuracy and reliability in feature extraction.

To generate the expression representation z, as shown in Figure 4.5, external attention [83] is applied to focus on important features and implicitly learn correlations among all expression samples. Following symbols in Eq. (4.2), we first compute the attention map $A = QM_k^T$ by multiplying the query vector Q and the external learnable transposed key matrix $M_k \in \mathcal{R}^{S\times d}$. Q is projected from a feature map $F \in \mathcal{R}^{N\times d}$, where N is the size of feature elements, d and S are hyper-parameters. We normalize the attention map A and then multiply it with another external value matrix M_v. M_k and M_v are generated by additional linear layers, which can be optimized by back-propagation during training on the entire dataset. The feature map F_{out} is as follows:

$$F_{out} = Norm(QM_k^T)M_v. \tag{4.11}$$

In the end, we obtain the refined feature map with linear complexity $O(d \cdot S \cdot N)$, which is suitable for resource-constrained mobile devices.

Feature Alignment for Domain Adaptation

Reasonably, we assume that samples with the same label are closer to each other in the feature space, while samples from different categories are farther apart, no matter which domain they come from. With the augmented dataset, pseudo labels, and the attention-based learning model, we design contrastive learning to minimize the domain discrepancy by aligning facial expression features between the training and testing datasets.

Specifically, given an acoustic facial expression sample x_s from the source domain, and a sample x_t from the target domain, we minimize the distance between x_s and x_t if two samples are from the same class while maximizing the distance between two samples from different classes. The output of the model is domain-independent expression representations. Following the supervised

Algorithm 6: Contrastive Attention-based Cross Domain Acoustic Expression Representation Learning

```
Input: source dataset $D_s$, target dataset $D_t$, epoch $E$, iterations $K$ per epoch, weight $\lambda$,
       contrastive attention-based model $f$
Output: source and target representations $Z^s$ and $Z^t$
1  for $e = 1$ to $E$ do
2      Calculate centroids in target domain using Eq. 4.9
3      Update pseudo labels for target data using Eq. 4.10
4      for $k = 1$ to $K$ do
5          for each batch do
6              Extract features with $f$ based on external attention in Eq. 4.11
7              Compute $L_{ce}$ for each batch from $D_s$
8              Compute $L_c$ from $D_s$ and $D_t$ using Eq. 4.12
9              Compute $\mathcal{L}_{ce}(\theta; D_s) + \lambda\mathcal{L}_{\mathrm{c}}^{t}(\theta; D_s, D_t)$
10         end
11         Back-propagate and update $\theta$ of model $f$
12     end
13 end
14 for each batch $X_{batch}$ do
15     Generate source domain expression representation $Z^s_{batch} = f(X^s_{batch})$ for $D_s$
16     Generate target domain expression representation $Z^t_{batch} = f(X^t_{batch})$ for $D_t$
17 end
18 return $Z^s$ and $Z^t$
```

contrastive loss in Eq. (4.6), we define the domain adaptation contrastive loss as follows:

$$\mathcal{L}_{\mathrm{c}}^{t} = \sum_{i \in I_t} \frac{-1}{|P_s(y_t^i)|} \sum_{p \in P_s(y_t^i)} log \frac{exp(z_t^i \cdot z_s^p/\tau)}{\sum_{a \in I_s} exp(z_t^i \cdot z_s^a/\tau)}, \tag{4.12}$$

where I_t denotes the set of target samples in a batch, I_s is the set of source samples, and $P_s(y_t^i)$ is a set of indices of all positive samples from the source domain. A positive sample means the label of the sample is the same as the pseudo label of the target anchor sample x_t. The domain adaptation contrastive loss aligns the expression representation in the target domain to the source domain. Finally, we define the acoustic expression representation learning loss function as:

$$\arg\min_{\theta} \mathcal{L}_{ce}(\theta; D_s) + \lambda\mathcal{L}_{\mathrm{c}}^{t}(\theta; D_s, D_t), \tag{4.13}$$

where $\mathcal{L}_{ce}$ is the standard cross-entropy loss applied on the D_s, λ is used to balance the two loss terms (0.5 in our experiments), and θ represents the model parameters.

Overall, the proposed contrastive external attention-based acoustic expression representation learning model is presented in Algorithm 6. After data augmentation, in each epoch, we first generate pseudo labels for target domain acoustic samples (lines 2-3). Then, we minimize the loss and back-propagate to update the model f (lines 4-12). After training, model f can align features for domain adaptation, and in turn minimize the distribution shift. We use the trained model f to generate the acoustic expression representations Z^s and Z^t (lines 14-17). Finally, we can train a classifier on source domain representations Z^s, and generate labels for the target domain representations Z^t. The proposed algorithm effectively enhances the expression recognition model performance across different users.

4.4: Evaluation and Discussion

We introduce the data collection process, the software, and the hardware setup for acoustic expression recognition implementation. Then, we evaluate the impacts of multiple factors (*e.g.*, location, time, people, mask) on the performance of FacER for expression recognition.

4.4.1: Data Collection

We recruited 20 volunteers (16 males and 4 females) to participate in the acoustic facial expression data collection process. To guarantee the heterogeneity of collected facial expressions, the recruited volunteers have different skin colors and grow up in different regions of the world such as Asia, North America, and Europe. Their ages range from 20 to 38. The participants are allowed to wear glasses, a hat, and other accessories during the data collection process. To simulate different scenarios in real-life, we collect data at different locations (*e.g.*, office, dining hall, garden) with different background noise. For example, we collect data in an office with people having conversations, online meetings, and computer alarm beeping.

We show the six facial expressions: anger, disgust, fear, happiness, sadness, and surprise as

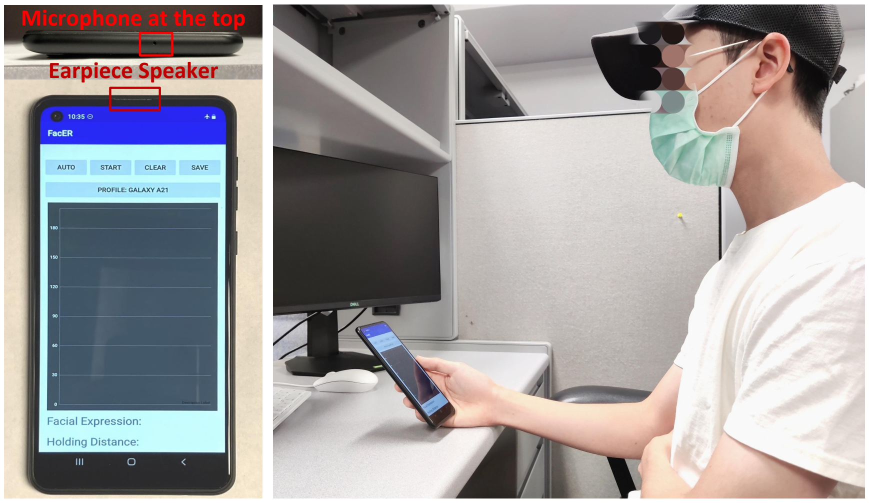


Figure 4.6: A smartphone and a volunteer for data collection.

examples at the beginning of data collection. Then, a volunteer starts with a poker face and performs their preferred styles of six expressions accordingly. Figure 4.6 shows the example of a smartphone and a volunteer during the data collection process. The volunteers are allowed to hold the phones in their most comfortable gesture while watching the smartphone screen. Considering the necessity of wearing masks during the COVID-19 pandemic, we also consider expression recognition in scenarios where the participants are wearing disposable masks. For each expression, we collect about 5 seconds when the participants wear masks, and another 5 seconds when they do not.

The acoustic facial expression collection process repeats 10 times for each expression per person with rest breaks during the data collection process. An independent observer helps record the label of each acoustic expression sample as the ground truth. The whole data collection process takes about a week intermittently. The collected dataset is over 1 GB in plain text format. Finally, we get 20,535 samples with the sliding windows segmentation, of which the window size is 0.25s. The design consideration is that if the window size is too small, it will be hard to reflect the integral process of facial muscle movements. If the window size is too large, it will easily mix instantaneous variations of different facial expressions.

4.4.2: Experimental Setup

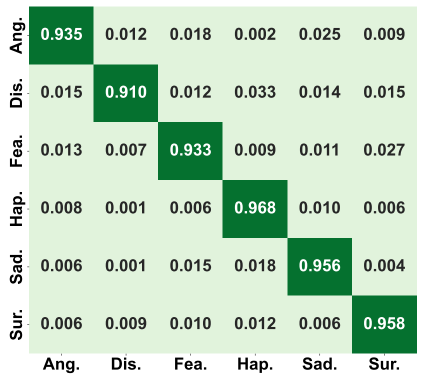
We use two Android phones: the Samsung Galaxy A21, and the OnePlus 8T to collect the acoustic signals. We develop the app to emit and collect signals based on the LibAS [204] and Chaperone [33], which provide a framework for acoustic sensing applications. LibAS can simplify the necessary signal processing procedures such as synchronization, which determines the start position of the sent signals in the received audio. The sensing signal is a chirp signal modulated from 19-23 kHz, which is emitted from the earpiece speaker of a smartphone.

We employ the SciPy library for signal processing. We use ResNet-18 [88] as the backbone model for our contrastive attention-based expression recognition model, which is implemented with Pytorch. We train the model on TensorEX Ubuntu 20.04 Server with 256GB DDR4 memory, Intel(R) Xeon(R) Gold 5218R 2.10GHz CPUs, and RTX A6000 GPUs.

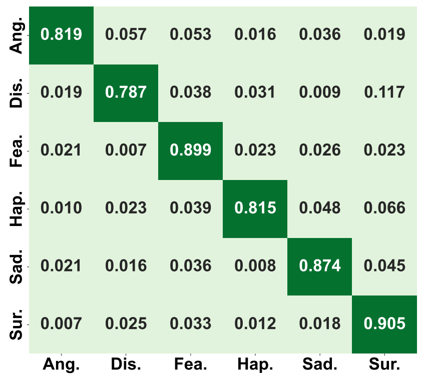
4.4.3: User Dependent Evaluation

The confusion matrixes are shown in Figure 4.7. Case 1: training and testing on the mixed dataset; Case 2: training on men's dataset and testing on women's dataset; Case 3: training on users with masks and testing on users without masks; and Case 4: training on users without masks and testing on users with masks. The tick labels are angry, disgust, fear, happiness, sadness, and surprise. The x-axis contains the prediction labels, and the y-axis contains the ground truth labels.

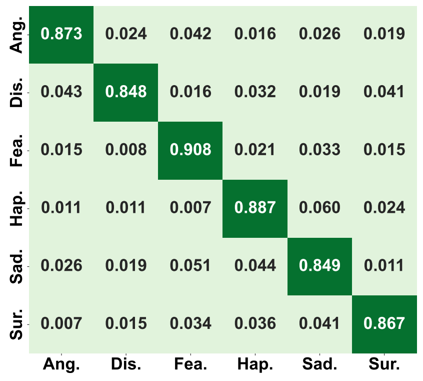
Case 1a. We first consider a simple scenario, where we can collect and label acoustic data from a group of users. Our goal is to recognize the facial expressions from this specific group of users, namely *mix testing*. We split 80% of the whole dataset as the training set and the remaining 20% as the testing set. We only use the external attention-based ResNet-18 model without cross-domain adaptation to implement classification. We use the SGD with a momentum of 0.9. The learning rate is 0.1 with a learning rate decay rate of 0.01. We show the performance of FacER with an accuracy heat map in Figure 4.7a. As we can see, the model can recognize each acoustic facial expression with more than 91% accuracy. Besides, we implement the 10-fold cross-validation, and the average testing accuracy is 94.3% with a standard deviation of 0.4%. However, the results are



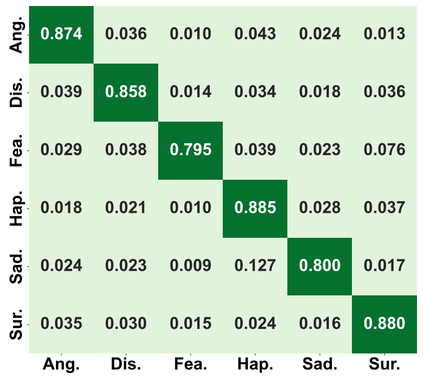
(a) Case 1: Mix testing.



(b) Case 2: Testing on Women's.



(c) Case 3: Testing on Mask's.



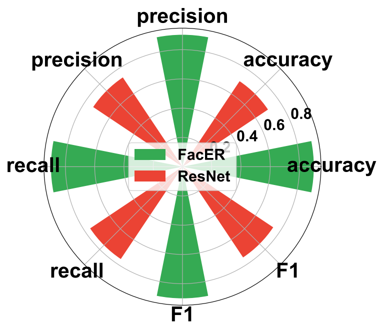
(d) Case 4: Testing on No Mask's.

Figure 4.7: Evaluations of different cases.

unsurprising because the training dataset and testing dataset belong to the same distribution. As a result, the model can easily fit in the dataset.

Case 1b. Next, we consider the impact of the environmental factors on the performance of FacER. We test the model performance on the data collected in three different locations, i.e., office, dining hall, and garden. We repeat the *leave-one-place-out* evaluation by training on data from two places and testing on data from another place. We implement Algorithm 6 to extract feature representations, and then we use a linear classifier with two linear layers network with hidden layer sizes 1,024 and 256 to implement expression classification.

We compare the performance of FacER with two baselines: (i) ResNet-18 [88], which is our backbone model trained with cross-entropy loss without adopting attention; (ii) XHAR [258], which is the adversarial training-based domain adaptation method for human activity recognition. The adversarial training is another major direction for cross-domain adaptation, so we choose it as



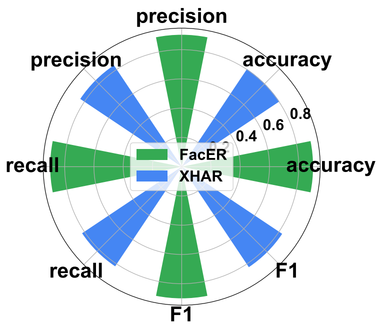


Figure 4.8: The bars of location factor evaluation.

the baseline. We adapt XHAR to make it applicable for facial expression recognition.

We report the average accuracy, precision, recall, and F1 values across three locations in Figure 4.8, which is a bar plot on the polar axis. FacER achieves 89.8% average accuracy, which is a bit lower than the mix testing method (94.3%). The standard deviations of the accuracy of FacER, ResNet, and XHAR from three leave-one-place-out evaluations are 0.017, 0.014, and 0.016, respectively. The results show that the location causes the distribution shift and affects the model performance because of the noise caused by different obstacles. Nevertheless, the model still achieves high performance with an F1 value of 90%, which is higher than the XHAR method (82.6%). The result proves that FacER can learn consistent and robust acoustic facial expression features even in a noisy environment with various types of noises.

Case 1c. We consider a more complex scenario that is related to time variation. The consideration is that the same facial expression may not be consistent over time. For example, sometimes people may show a wide smile while other times people may show a gentle smile to express happiness. Therefore, to evaluate the time factor for acoustic facial expression recognition, we split the dataset into two subsets according to the time. The first subset is the first 8 times of data collection for each facial expression per person, which is the training dataset. The second subset is the last 2 times of data collection, which forms the testing dataset.

The results show that the time factor affects the model performance. Under the time factor condition, the accuracy of FacER is 90.3%. The precision and recall are 90.5% and 90.3%, respectively. The model has slightly worse performance (4% lower accuracy) than Case 1a, which

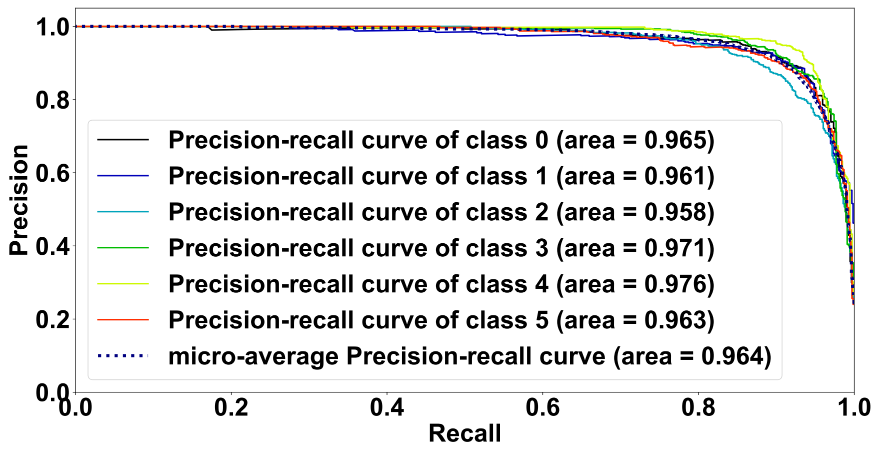


Figure 4.9: The precision-recall curve of time factor evaluation.

shows that the changes of the same expression over time indeed degrade the recognition accuracy. Nevertheless, FacER can still achieve high performance, which shows the efficacy of the designed model in acoustic expression recognition. We show the precision-recall score curve in Figure 4.9. Precision is $\frac{tp}{tp+fp}$ and recall is $\frac{tp}{tp+fn}$, where tp means true positive, fp means false positive, fn means false negative. The tradeoff between precision and recall for various thresholds is depicted by the precision-recall curve. The area in Figure 4.9 is calculated by the average precision (AP): $AP = \sum_n (R_n - R_{n-1})P_n$ where P_n and R_n are the precision and recall at the n_{th} threshold automatically set by Scikit-plot [152]. For different classes, FacER can achieve at least a 0.958 AP score. A high precision-recall (area) score under the curve shows both high recall and high precision, which indicates low false-positive and false-negative scores.

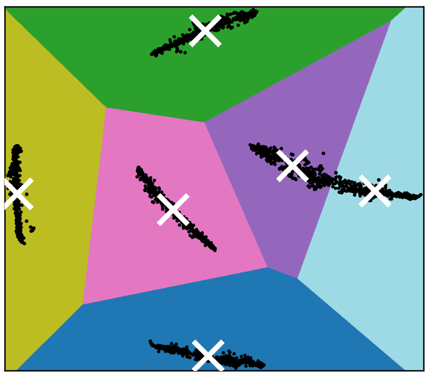
4.4.4: User Independent Evaluation

Case 2. Now we consider a more general scenario where the model is trained and tested on different sets of users. Different people have different face geometries and behaviors. As a result, they may produce distinct feature patterns. To evaluate FacER in the user-independent scenario, we use the dataset from 16 men as the training dataset and that from 4 women as the testing dataset.

We first show the efficacy of the K-means model in clustering the acoustic facial expression representations. We randomly select 2,048 samples from the training dataset in Figure 4.10a and from the testing dataset in Figure 4.10b, respectively. As we can see, the K-means model can separate the six types of expressions from both training and testing sets, which indicates the learned representations contain consistent and distinctive features. During the iterative training process, the pseudo labels of the testing data can be accurately updated. Therefore, the proposed model can effectively compose positive and negative pairs for contrastive attention learning.

The accuracy heatmap is shown in Figure 4.7b, and the average accuracy is 85%, which is lower than the results from Case 1a because of the user expression differences. We notice that men and women have different ways of expressing "disgust", which causes FacER to only achieve 78.7% accuracy. But the result shows that men and women have high similarity in expressing "surprise", which achieves 90.5% accuracy.

To evaluate the efficacy of the domain adaptation of FacER, we compare FacER with the three baselines: XHAR [258], SonicFace [75], and ResNet [88] as shown in Figure 4.11. XHAR and ResNet are set up as in Case 1b. SonicFace uses both FMCW and pure tone signals to extract different features. It focuses on tracking the movement of facial components such as the eye, eyebrow, and cheek, and it uses 1D convolution for signal processing. In contrast, we regard the spectrogram of the received echos as an image, which is the instantaneous static facial expression. Different expressions will generate different spectrogram features as the different pixels of facial expression images. Therefore, we use the 2D convolution. Due to the key difference in the applied signals, we directly report their best performance in Figure 4.11. For SonicFace, the average accuracy of the intra-session classification is 78.6%, while the best accuracy of the user-independent model with calibration is 72%. For the user-independent Case 2 scenario, FacER can achieve 85% accuracy and 85.2% F1 value. The adversarial training-based XHAR method only achieves 79.1% accuracy and 78.7% F1 value. The results show the superior performance of our proposed contrastive external attention-based representation learning method, as it helps extract robust and accurate acoustic facial expression features.


(a) Clustering on the training set.

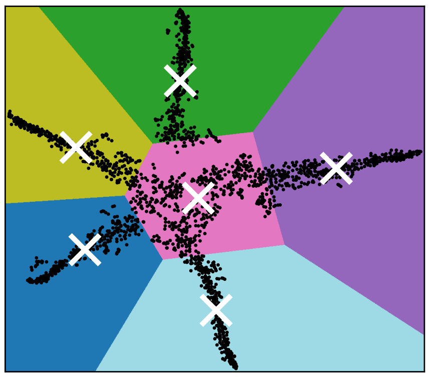

(b) Clustering on the testing set.

Figure 4.10: K-means clustering on 2,048 sampled representations, which are processed with TSNE dimension reduction.

4.4.5: Mask Factor Evaluation

Case 3 and 4. Masks are obvious obstacles to camera-based facial expression recognition methods. It is very common for people to wear masks during the Covid-19 pandemic. We investigate the performance of FacER when people wear masks. As mentioned before, half of the dataset is from volunteers wearing masks (mask dataset), and the remaining half is from volunteers without wearing masks (plain dataset). We train FacER on the plain dataset and test on the mask dataset, and the result is shown in Figure 4.7c. The average accuracy is 87.2%. Then, we train FacER on the mask dataset and test on the plain dataset, and the result is shown in Figure 4.7d. The average accuracy is 84.9%. As we can see, the mask has a noticeable impact on the performance of FacER.

Another interesting finding is that FacER achieves the highest accuracy of 90.8% about the "fear" expression when training on the plain dataset, while it achieves the lowest accuracy of 79.5% about "fear" when training on the mask dataset. This is because the mask itself will generate some echoes back to the microphone, which will cloud the echoes from the facial expressions. It is harder for FacER to learn robust "fear" features when people wear masks. Nevertheless, for the "happiness" expression, people have a larger facial muscle movement, which could trigger the movement of masks. As a result, it can achieve 88.7% and 88.5% accuracy when testing on the mask and plain datasets, respectively. Overall, in different mask factor scenarios, FacER can

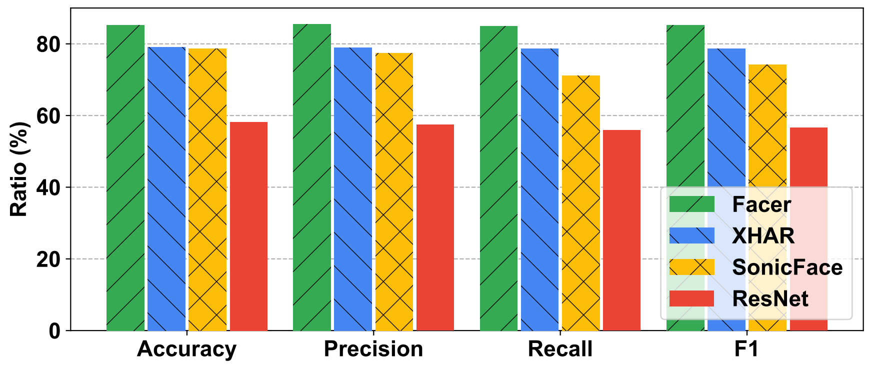


Figure 4.11: The comparison across different models.

achieve comparable accuracy with Case 2. The results confirm the cross-domain adaptation ability of our proposed contrastive attention-based representation learning method.

4.4.6: Discussion

Data Perspective

FacER is a data-driven acoustic facial expression recognition system based on the contrastive attention-based deep learning model. The quality and quantity of the acoustic sensing data are important to the performance of FacER. We will collect more data from female users and underrepresented groups to reflect the diversity of facial expressions. We will also consider other forms of emotional expressions such as hand gestures. Moreover, we will collect data on facial expressions that gap a longer time such as after one week for the time factor evaluation. We will also try applying generative models to synthesize more data.

Model Perspective

In this chapter, we mainly consider the scenarios where the users hold the phone at a distance of 20-50 cm. In the future, we will investigate the performance of FacER when the distance between the user and the phone is longer. We will also study how to determine the start and end of an expression, so we can effectively segment acoustic signals to extract facial expressions for inference.

System Perspective

We understand the design difference between different phone hardware, especially its impact on the direct path from the speaker to the microphone. We will evaluate the effect of different smartphones on ultrasound signal transmission in the future. Besides, many other real-world impact factors such as slight hand shaking, finger swiping on the phone, and the orientation of a phone should be taken into consideration. We will continue optimizing resource usage on mobile devices and mitigate the security and privacy concerns of mobile sensing data.

4.5: Summary

Facial expression recognition, as a universal way to understand human emotions, is significant not only for content recommendation but also for mental health care. In this chapter, we designed FacER, an acoustic facial expression recognition system using a smartphone earpiece speaker and microphone. To enhance the accuracy and robustness of FacER, we proposed a contrastive external attention-based representation learning model to learn robust facial expression features across different users in various noisy scenarios. Real-world experiments show that FacER achieves expression recognition with more than 85% accuracy even when the users are wearing a mask, a new norm during the COVID-19 pandemic. In the future, we will explore more expression variations such as emotional hand gestures to better capture users' emotions in a privacy-preserving manner. We will continue to improve the fairness of the facial expression recognition system, i.e., the training data should be diverse enough such that the model would not be biased. Moreover, we will optimize resource consumption on mobile devices.

CHAPTER 5: MULTI-GRAINED ACTIVITY SENSING INFORMATION RELEASE

5.1: Introduction

Smartphones and wearable devices have seamlessly integrated into our daily lives, offering personalized services with an array of motion and position sensors. These sensors are adept at capturing three-dimensional movements and orientations, enabling continuous monitoring of user activities. Such sensors are categorized by operating systems as low-risk, allowing apps to access their data without specific permissions. For example, on Android platforms, an application can retrieve sensor data at rates over 200 Hz without needing direct approval from the user [50]. Consequently, an application designed for navigation, utilizing these sensors to determine compass direction, can continuously gather activity-sensing data, highlighting the ubiquitous presence of these mobile devices in our daily routines.

The data harvested by mobile sensors often exceeds what is necessary for the intended functionality of user applications, leading to apps gaining excessive access to this information. This surplus of access has made mobile applications overly privileged in terms of the data they can collect [201, 237], containing superfluous or minimizable details that service providers do not necessarily need. Moreover, there is a risk that this data, which is more extensive than needed, could be exchanged or compromised by entities other than the service provider [11]. Two primary concerns arise from this overprivileged access in mobile activity sensing: (i) *Metadata-level overprivileged issue*, where applications can determine sensitive user information such as age or gender. The data on physical activities, enriched with unnecessary personal details, could lead to privacy breaches, as individuals exhibit unique activity patterns based on their characteristics. These distinct patterns could potentially be leveraged by malicious attackers to deduce sensitive user information [100, 101, 141].

(ii) *Feature-level overprivileged issue*: This issue arises when apps access more detailed activity data than necessary, potentially exposing users' sensitive behaviors. For example, consider an elderly individual using a smartwatch primarily for fall detection purposes [182]. While the device's main function is to ensure safety, it can also monitor detailed hand movements [37, 110, 147, 212]. This capability might allow a nefarious entity to decipher users' banking passwords or other sensitive information by analyzing precise sensor data [44, 184, 187]. In essence, app developers with malicious intent could exploit the excess motion data to uncover private behaviors and information unrelated to the app's intended use [192]. Thus, sensors deemed low-risk might pose significant privacy threats, as the detailed data they collect could inadvertently reveal confidential user information.

Many efforts have been made to address overprivileged issues in mobile activity sensing data. For metadata-level overprivileged concerns, adversarial training techniques [21, 115, 133] have been employed to obscure personal attributes within sensor data. When it comes to feature-level issues, a range of filtering strategies [20, 30, 79, 171] has been developed to eliminate unnecessary raw data, leveraging predefined private activity patterns from users. For example, MaskIt [79] offers a mechanism to decide if a data segment should be disclosed or concealed. Nonetheless, these approaches typically necessitate gathering labeled private data for model training or specific patterns for filtering, which is infeasible in real-world settings. Besides, semantic label-based filtering is generally coarse-grained, opting to either fully expose or suppress data segments, thereby destroying the utility of the data when a segment is associated with a private label.

Adding noise to data [87, 150, 219] is another common method for protecting privacy, aiming to distort both the private attributes and activity semantics. However, due to the repetitive patterns in sensor data over time, substantial noise levels must be introduced to prevent data cleaning techniques from effectively removing the noise, which can significantly impair the utility of the data, particularly in recognizing activity patterns [143]. Thus, noise addition approaches tend to be coarse-grained, often impacting the granularity required to finely mitigate overprivileged issues without compromising data utility.

To address the aforementioned challenges, we introduce Hippo, a system built on a generative AI model that reconstructs mobile activity sensing data at multiple granularity levels. The primary objective of Hippo is to obscure sensitive attribute details and simplify overly detailed activity features that are not essential. Hippo employs a diffusion model, a type of generative model, to infuse noise into the sensor data with a learning-based diffusion process. This involves a two-step process: first, introducing noise to the data through forward diffusion, and then learning to refine the data by removing the noise during reverse diffusion. These steps are integral to the diffusion model's training phase, ensuring that while the noise disrupts sensitive attributes, the essential activity features are preserved, maintaining the utility of the data.

To refine activity feature representation, Hippo does not merely inject noise; instead, it crafts multi-grained data by pruning features that are not intended to be released by users. This approach is grounded in the understanding that human activities exhibit a hierarchical semantic structure, where activities can be broken down into basic actions or aggregated into broader activity categories. For instance, the action of "jumping" comprises basic movements such as "arms swinging", "legs contracting", and "stretching", and can be categorized under the broader term "locomotion" along with actions such as "running".

In its operation, Hippo leverages stacked convolutional autoencoders (CAEs) to derive multi-resolution feature representations from data segments. This capability stems from the convolution and pooling operations and the architecture of deep neural networks (DNNs), which are known to capture data representations at varying levels of granularity. Research has shown that different layers in a DNN encapsulate representations at diverse scales and depths, establishing a connection between these representations and semantic concepts. Following this, Hippo utilizes these multi-resolution representations to produce multi-grained sensing data, employing the diffusion model's generative capabilities. This is particularly crucial for applications such as travel or pedometer apps that depend on raw data rather than processed feature representations, necessitating a method to regenerate data that aligns with the original yet without the sensitive or unnecessary details.

Hippo plays a critical role in data reconstruction, focusing on two main aspects: perturbing personal attributes and generating multi-grained data. In terms of personal attribute perturbation, Hippo effectively obscures potential attributes or metadata, which are not essential for the functionality of activity sensing applications, thereby enhancing user privacy [21, 133, 140, 173]. When it comes to generating multi-grained data, Hippo allows users or applications to define the desired level of data granularity, which corresponds to the number of stacked convolutional autoencoders (CAEs) used. This approach eliminates the need for predefined privacy patterns or labels during data reconstruction.

The efficacy of Hippo is assessed by examining the multi-grained sensor data across various activity recognition and attribute inference models. In the realm of metadata-level security, Hippo manages to decrease the risk of private attribute exposure by approximately 50% while maintaining the accuracy of activity recognition akin to that of raw data. For feature-level security, the reconstructed multi-grained data exhibit varied levels of activity recognition precision. Furthermore, we demonstrate the utility of reconstructed data in fall detection and pedometer services, showcasing how users can manipulate data granularity to safeguard their privacy.

The main contributions of Hippo are summarized as follows:

- We introduce an innovative approach that utilizes a noise diffusion process to obscure personal attributes effectively. This method maintains the integrity of activity recognition performance while eliminating the need for users to provide specific private data or labels.
- We develop a hierarchical latent feature guidance diffusion model aimed at generating multi-grained data. This design allows for precise control over the dissemination of information, addressing and reducing feature-level overprivileged concerns.
- Through both theoretical analysis and practical experimentation, we demonstrate that the comprehensive model Hippo is equipped to tackle overprivileged issues at both the metadata and feature levels across a range of different contexts.

5.2: Problem Definition

In this section, we formally define and analyze the activity sensing utility and privacy from the semantic perspective.

5.2.1: A Motivating Example

We assume a strong attacker who can train various attribute inference and activity recognition models offline to recognize personal attributes and activity labels. The attacker can implement the re-identification attack [84]. Specifically, the attacker knows the defense methods and has a dataset with private attribute labels. Then, they can build a sanitized version of the dataset by passing the dataset through the defense methods. After that, the attacker can train a new private attribute identification model based on the sanitized datasets. Considering many apps need raw sensor data instead of activity labels, Hippo reconstructs raw sensor data to filter out sensitive information. Users should have full control over the mobile sensing data that they share, but the current OS does not provide any permission restriction. Currently, for example, Android provides eight types of motion sensor APIs and six types of position sensor APIs to access raw sensor data [51]. To protect activity-sensing information, we assume attackers can only access sensor data after being processed by Hippo. The mobile sensors can be designated solely for the secure realm (e.g., ARM TrustZone [48]) as in prior research [65], and then processed by Hippo.

5.2.2: Mobile Activity Sensing Data Utility

Human activities inherently exhibit a hierarchical organization, essential for identifying and understanding activities at various levels of granularity [130]. For instance, activities such as walking, running, and jumping fall under the broader category of locomotion, as illustrated in Figure 5.1. Initially, we delineate the concept of granularity within the realm of activity semantics.

Definition 1 *Let A represent a set of activity semantics, and the granularity G is a function that maps each activity $a_i \in A$ to a level of detail L, where $L = \{l_1, l_2, \cdots, l_n\}$ with l_1 representing the most detailed level and l_n representing the broadest level of categorization. For any two levels l_i*

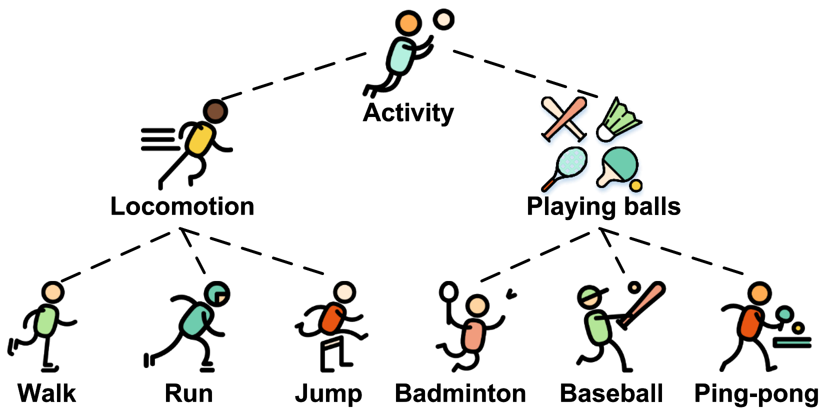


Figure 5.1: The hierarchical semantic nature of activities.

and l_j, if $i < j$, l_j is more general than l_i.

Each activity label a_i is associated with a level of granularity l_k, which indicates the categorization applied to the activity. Consider a segment of the time series $X = (row_1, row_2, \cdots, row_w)$, where row_i has multiple values such as the 9-axis values of inertial sensors, and w is the window length of X. The semantic label of X is obtained by the activity recognition models such as DNN models [34, 217] and statistical methods [113, 138]. If the fine-grained activity features in X are removed, then X can only be recognized in coarse-grained activity semantics. The role of Hippo is to generate multi-grained data by removing fine-grained features. For generated data X_r at granularity level l_k, X_r can be recognized into multiple possible fine-grained activities, which compose a hierarchical structure.

We define activity recognition data utility function as $U(a, \hat{a}) = \frac{U_{\hat{a}}}{U_a}$, where U_a is the activity recognition accuracy score on raw data a and $U_{\hat{a}}$ is the accuracy score on reconstructed data $\hat{a}$. The accuracy score is computed by an activity recognition model. $U(a, \hat{a})$ quantifies the utility of data after generalizing activity semantics from a to $\hat{a}$. For example, the raw sensor data contains the highest-resolution features corresponding to fine-grained activity semantics. The maximum utility is 1 if we keep the raw data. The $U(\cdot)$ monotonically decreases when the semantic of X is generalized from a fine-grained $\hat{a}_{i-1}$ to a coarse-grained $\hat{a}_i$:

$$0 < U(a, \hat{a}_i) \leq U(a, \hat{a}_{i-1}). \tag{5.1}$$

The utility of sensing data is intrinsically linked to the specific functions of the applications using it. While it is presumed that applications transparently declare their data needs for intended functionalities, there exists a risk that the data gathered may contain excess information, vulnerable to unauthorized access or misuse through third-party transactions or breaches. Therefore, the users can determine the granularity level of the reconstructed data to generate coarse-grained data, so the mobile sensing data can be recognized into ℓ different activities. Or the user can utilize Hippo to just perturb the metadata information linked to the activity sensing data while retaining the raw activity sensing features.

To address the issue of data being overprivileged, both service providers and users can engage in a cooperative effort to determine the i-th layer feature guidance for data reconstruction. The i is related to the number of stacked convolutional autoencoders (CAEs) for multi-resolution extraction. For instance, if the raw sequence data X_{arm} is indicative of "arm swinging" within bodily gestures, then X_{arm} comprises high-resolution features capable of identifying the "arm swinging" movement. For a given layer i, Hippo enhances privacy by stripping away the detailed "arm swinging" feature during the reconstruction of data X_r, utilizing the guidance from the i-th layer's features extracted by stacked CAEs. If $i = 0$, then Hippo only perturbs metadata information linked to the sensing data. If $i > 1$, the Hippo utilizes the i-th layer to guide the data reconstruction process. This i-th layer represents a more abstract view, omitting the detailed "arm swinging" feature.

As a result, the reconstructed data X_r is categorized under broader, less specific activity semantics such as "jumping" or "running" without the granularity to distinguish "arm swinging". Similarly, if the reconstruction process generalizes the features corresponding to "jumping", the activity could be classified under the even more generic category of "locomotion". The selection of layer i is tailored to meet the demands of the particular application, determined by setting the requisite number of CAEs for extracting features at different resolutions. This process ensures that the reconstructed data aligns with the necessary level of detail dictated by the application's requirements, balancing the need for functionality with the imperative for privacy.

5.2.3: Sensing Data Overprivileged Information

In the context of mobile activity sensing, we categorize overprivileged information as any data that is irrelevant to the app's specific task. For such applications, metadata, which includes personal attributes extracted from activity sensing data, is considered ancillary information [21, 133, 140]. This metadata is not essential for the app's primary functions and poses potential risks for privacy breaches. To address this, Hippo is designed to indiscriminately distort this metadata within the raw data, eliminating the need for users or developers to explicitly identify which attributes are private. This approach ensures that sensitive personal details are obscured, reducing the likelihood of unintended privacy exposure.

In addition, raw data may contain fine-grained features that are unnecessary for applications. To generalize unnecessary activity semantics, Hippo reconstructs multi-grained data, allowing users to specify the desired granularity of data. Given an activity semantic distribution D of raw sensing data X and activity distribution G of reconstructed data X_r, we define mutual entropy $H(G; D)$ to measure the amount of overprivileged information:

$$H(G; D) = H(G) - H(D), \tag{5.2}$$

where $H(\cdot)$ denotes the Shannon entropy. $H(D)$ and $H(G)$ are related to the possible semantics of X and X_r, which are calculated based on the classification prediction score. The lower bound of $H(D)$ is 0 if the score is 1. We define the threshold Δ to control the amount of overprivileged information in reconstructed data. Hippo generalizes the raw data X to increase the entropy $H(G)$, so as to guarantee $H(G; D) \geq \Delta$, where $0 \leq \Delta < 1$. A smaller Δ means more overprivileged information. When $\Delta = 0$, the activity data X_r is kept as the original sensor data X. The lower bound on $H(G)$ is calculated by $H(D)$ and the lower bound Δ on $H(G; D)$.

5.3: System Design

In this section, we first introduce preliminary knowledge and the system model. Then, we provide a detailed design and analysis of Hippo for deep hierarchical data generation.

5.3.1: Background Knowledge

Stacked Convolutional Autoencoder Model

Convolutional autoencoder (CAE) [145] is widely used to extract features in an unsupervised way. The convolution module has local undirected connections and shared parameters between the input layer and the hidden layer. In the encoder part, the input X is encoded into i-th hidden unit z, which is given by $z_i = \sigma(XW_i + b)$. The σ is an activation function, W_i is the learned weights and b is the bias. In the decoder part, hidden unit z_i is decoded back to reconstructed input X' with a deconvolution module, which is given by $X' = \sigma(z_i\tilde{W}_i + d)$. where $\tilde{W}$ is the flipped version of W and d is the bias.

The stacked convolutional autoencoder consists of multiple CAEs where the input of each hidden layer is the output of the previous layer. Each data vector $\mathbf{X}^m$ is translated into another vector $\mathbf{X}^n$ in a lower dimensional space where $m > n$. The pooling module such as max pooling and mean pooling makes feature representations invariant to the small changes in the input. Unpooling is used in the decoder part which uses the recorded locations of activations in the pooling operation to place the reconstructions. The loss function is to minimize the mean square error between input X and reconstructed input X':

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - X_i')^2. \tag{5.3}$$

Diffusion Model

The diffusion model (DM) is a form of generative model. Previous studies [90, 191, 193] have demonstrated that DM produces data of higher quality compared to existing generative models (e.g., GAN [77]). During the training phase, the DM introduces random noise $\epsilon_t \sim \mathcal{N}(0, \sigma_t^2)$ to input data using a noise scheduler, known as the *forward diffusion process*. Specifically, given a real data distribution $q(x)$ and a data sample x_0 drawn from $q(x)$, noisy samples $x_1, \cdots, x_T$ are generated by adding Gaussian noise to the output of the previous step. This process involves T steps (e.g., 1000) following a Markov chain. The conditional distribution is:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \tag{5.4}$$

where $\beta_t \in (0, 1)$ denotes a variance schedule controlling the step size. There are many methods to set up β_t. For example, we can use a linear function to map a beta range (*e.g.*, [0.0001, 0.1]) to a sequence of β. With the increasing step t, the data sample x_0 will lose the distinguishable features because of the added noise. Suppose $\alpha_t = 1-\beta_t$, $\bar{\alpha}_t = \Pi_{i=1}^{t}\alpha_i$, noise $\epsilon(x_t, t)$ is the merged multiple Gaussians $\mathcal{N}\ (0, \sigma^2 I)$, then

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon(x_t, t). \tag{5.5}$$

The posterior mean μ_t in the forward process is as follows:

$$\mu_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon(x_t, t)). \tag{5.6}$$

Then, the diffusion model p_θ learns to remove noise from x_t, called the *reverse diffusion process*. Following the existing work [53, 90, 193], we use the UNet [176] as the backbone architecture to learn to remove noise. The input of UNet is a noised x_t, and the output is the added noise ϵ or clear data x_0. The UNet includes a stack of downsampling convolutions followed by upsampling convolutions. The advantage is that UNet can progressively lower the feature map resolution and then increase the resolution in the diffusion process. Besides, UNet contains the skip connection to add details across layers with the same spatial size. Specifically, p_θ learns to approximate the conditional probabilities $p(x_{t-1}|x_t)$.

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \tag{5.7}$$

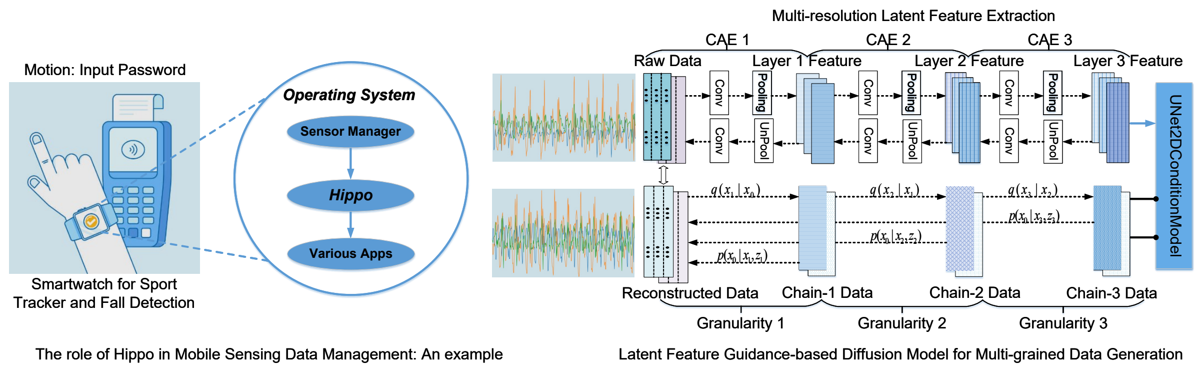


Figure 5.2: The latent feature guidance diffusion model of Hippo, which acts as a middleware between the OS Sensor Manager and Apps.

where $\Sigma_\theta(x_t, t))$ is set as $\sigma_t^2 I$ to time-dependent constants [90]. Eq. 5.7 can be reparameterized to predict ϵ_t from input x_t as follows:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)), \tag{5.8}$$

where $\epsilon_\theta(x_t, t)$ is the predicted noise. Finally, by minimizing the differences between the forward process posterior mean μ_t and the reverse process mean μ_θ, the loss function can be simplified as

$$\begin{aligned} L = ||\mu_t - \mu_\theta|| &= \mathcal{E}_{t\sim[1,T],x_0,\epsilon_t}[||\epsilon_t - \epsilon_\theta(x_t, t)||^2] \\ &= \mathcal{E}_{t\sim[1,T],x_0,\epsilon_t}[||\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t)||^2]. \end{aligned} \tag{5.9}$$

In the sampling phase, we input random noise as a seed to the diffusion model, and the model can generate new data such as images or sensor data in the reverse diffusion process.

5.3.2: System Model

We focus on a system designed for mobile activity sensing applications, where Hippo operates as middleware on a smart device, as depicted in Figure 5.2. The raw sensor data X is collected from various sensors (e.g., accelerometer, gyroscope) of a mobile device over a defined time period (e.g., 1 second). X encompasses not only the desired information but also overprivileged data, and completely eliminating this data compromises its utility. To address this overprivileged data issue, Hippo reconstructs a data representation X_r at a specific granularity. X_r serves as a less privileged version of X, providing applications with the necessary information for their tasks, while restricting access solely to X_r instead of the raw data X.

To address metadata-level overprivileged issues, we employ a well-trained DM to perturb metadata within raw sensing data. Initially, the raw data undergoes the forward diffusion process to introduce noise. Subsequently, during the reverse diffusion process, Hippo employs a learning-based approach to eliminate the noise. However, the residual noise after the noise addition and denoising stages within the DM model can perturb the metadata in the sensing data. Nevertheless, the primary semantic information regarding raw activity is preserved, as it constitutes the principal features of the activity sensing data.

To address feature-level overprivileged issues, we leverage both the CAE model and the DM to produce multi-grained data. Initially, we extract multi-resolution features from a segment of data X using the stacked CAE model. Subsequently, employing a random noise seed and guided by the multi-resolution features, Hippo utilizes a well-trained DM to generate multi-grained data. This generated data retains specific-layer feature information while sacrificing finer-grained features. Consequently, the resulting data encompasses various granularities of activity semantics, as expressed by the multi-resolution feature representations. Users or applications can then specify the desired level of granularity for data reconstruction to alleviate the overprivileged issues.

To implement the functionalities described above, we train both the CAEs and DM, as illustrated in Figure 5.2, utilizing publicly available datasets. The core component of Hippo is the latent feature guidance-based diffusion model, which comprises two primary modules, depicted in Figure 5.2: (i) A self-supervised stacked convolutional autoencoder (CAE) responsible for extracting multi-resolution activity representations without necessitating private labels from users for model training (refer to Section 5.3.3); (ii) A hybrid diffusion model (DM), incorporating both conditional and unconditional aspects, based on multi-resolution representations to generate multi-grained activity sensing data (refer to Section 5.3.4).

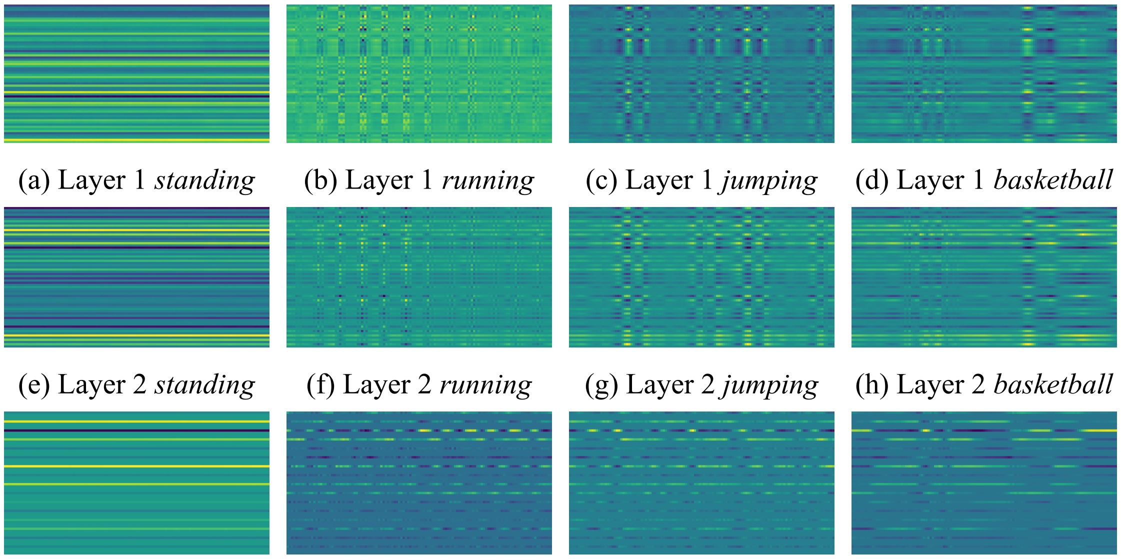
5.3.3: Multi-resolution Feature Extraction

We propose the utilization of a self-supervised stacked CAE to extract multi-resolution features, which serve as guidance for the generation of multi-grained data. The convolution module (CM) has been widely employed for feature extraction from time series data [74, 102, 243]. For instance, Lasagna [130] was introduced to produce hierarchical activity representations using CM. CM offers dual benefits for the utility of extracted features. Firstly, it can leverage local connectivity by focusing on multiple receptive fields of the input, which is particularly suitable for time series data where local patterns often hold more informative value than individual data points [243]. Secondly, CM implements weight sharing, implying that the same weight is applied across different filters. Each filter can capture patterns across various dimensions of the time series, ensuring that the model can recognize a pattern regardless of its position in the time series. Consequently, CM proves effective in extracting robust features for activity recognition.

We organize raw sensor data into a structured tensor for input into stacked CAEs, as depicted in Figure 5.2. Each CAE takes either the raw data X_i or the output feature z_i from the preceding CAE as input. The output of a CAE is denoted as feature z_j, where i and j represent the indices of stacked CAEs, with $0 < i < j$. For instance, the input of CAE-1 is the raw sensor data, and its output is the layer-1 feature. CAE-2 then takes the layer-1 feature as input. Normally, z_j resides in a lower-dimensional space compared to z_i. However, to maintain the same spatial dimensionality between z_i and z_j, we pad z_j with zeros to match the dimension of z_i.

The encoder of each CAE comprises a convolutional layer and a pooling layer. During the encoding process, these layers learn multi-resolution activity features. Conversely, the decoder consists of an unpooling layer and a deconvolution layer. Throughout decoding, these layers are employed to reconstruct the input data within each CAE. We train each CAE independently, aiming to minimize the mean square error between input data and reconstructed data, as expressed in Equation 5.3. Following the training process, we can utilize the encoder of stacked CAEs to extract multi-resolution activity features.

We select three activities from the DSA Dataset [14]: running, jumping, and playing basketball.



(a) Layer 1 *standing* (b) Layer 1 *running* (c) Layer 1 *jumping* (d) Layer 1 *basketball*

(e) Layer 2 *standing* (f) Layer 2 *running* (g) Layer 2 *jumping* (h) Layer 2 *basketball*

(i) Layer 3 *standing* (j) Layer 3 *running* (k) Layer 3 *jumping* (l) Layer 3 *basketball*

Figure 5.3: Visualization comparison of multi-resolution features extracted by stacked CAEs w.r.t. three activities.

These activities exhibit a hierarchical semantic structure, where playing basketball encompasses running and jumping. As illustrated in Figure 5.3, the layer 1 features of all three activities are discernible from each other. This disparity arises because the CAE is capable of extracting distinct feature components from the raw data.

As we progress to layer 2, the features of playing basketball and jumping start to resemble each other. This convergence can be attributed to the pooling and convolution modules learning a low-resolution version of the layer 1 features. Consequently, some intricate features specific to playing basketball and jumping become obscured, rendering the two activities indistinguishable. Similarly, the layer 3 features of running, jumping, and playing basketball exhibit similarity, as depicted in Figure 5.3. The visualization outcomes indicate that stacked CAEs can filter out fine-grained features as the depth increases. Therefore, by employing varying numbers of CAEs, Hippo can extract multi-resolution features encompassing different levels of information.

The generation of multi-resolution features is facilitated by the convolution and pooling layers within the self-supervised CAE. The convolutional layer is responsible for learning various aspects of features from the input data. Essentially, the input data is partitioned into multiple receptive fields for convolution. By employing different kernels for each receptive field, the CAE learns features from distinct parts of the input. Consequently, intricate feature maps of the input are obtained as a result. Previous studies in visual analytics have illustrated that the convolutional layer can discern different aspects of data, such as detecting edges and lines [17, 64, 67]. Moreover, deep convolutional neural networks have been shown to capture multi-grained feature representations [16, 174, 188].

Within each CAE, the pooling layer plays a crucial role in filtering out feature details by reducing spatial resolution in feature maps. Essentially acting as a bottleneck, the pooling layer compresses the features extracted from the input. This pooling operation limits the amount of information that can traverse through the neural network, compelling the model to focus on learning the most pertinent information while discarding irrelevant details.

Various resolutions of feature representations convey different levels of activity semantic information. For instance, in Figure 5.2, the raw data feature z_0 is considered as semantic granularity-0, encompassing the most detailed motion features. On the other hand, the layer-1 feature z_1 corresponds to semantic granularity-1, containing coarser motion information achieved by blurring features present in z_0. The convolutional and pooling operations contribute to z_1 losing detailed features compared to z_0. Furthermore, as the number of stacked CAEs increases, the information encapsulated in the upper layer features z_i diminishes.

Lemma 1 *Given a stacked two CAEs whose outputs are layer features z_i and z_j respectively. Then z_j is low-resolution version of z_i such that $\mathcal{H}(z_j) < \mathcal{H}(z_i)$ when $i < j$.*

We use the H-score $\mathcal{H}$ proposed in [95, 232] to quantify the informativeness of extracted features. $\mathcal{H}$ is computed in an information-theoretic framework to show that DNN extracts informative features for learning tasks. The $\mathcal{H}(\cdot)$ function is defined as follows:

$$\mathcal{H}(s) = \frac{1}{2}\mathbf{E}_{P_Y}[||\mathbf{E}_{P_{X|Y}}[\wedge_{\widetilde{s}}^{-1/2}\widetilde{s}(X)|Y]||^2]. \tag{5.10}$$

The $\mathcal{H}$ function takes as input the feature embedding matrix and label matrix of data samples. It quantifies the quality of features generated at any layer of the DNN. A higher value of $\mathcal{H}$ indicates that more information in the feature embeddings is relevant to the label. We compute the H-score on the feature embeddings generated from stacked CAEs using the DSA dataset [14]. The H-scores for layer-1 features, layer-2 features, and layer-3 features are 7.717, 7.518, and 7.423, respectively. Thus, using this framework, we demonstrate that stacked CAEs can extract multi-resolution features containing varying amounts of information. In Section 5.3.4, we devise the data generation model conditioned on z_i, which generates data containing multi-grained features.

5.3.4: Multi-grained Sensing Data Generation

Considering that many apps rely on raw data formats, Hippo is designed to generate multi-grained data to mitigate overprivileged information. The multi-resolution features extracted in Section 5.3.3 serve as the basis for multi-grained data generation. Suppose an app necessitates the i-th granularity of sensing data, corresponding to the i-th layer feature z_i in the multi-resolution feature extraction model. Leveraging the well-trained latent feature guidance diffusion model, Hippo generates the i-th granularity data from random noise conditioned on a specific resolution feature z_i.

During the offline training stage, the raw data x is encoded into a bipartite latent representation (x_T, z_i). The diffusion latent x_T is produced via the forward diffusion process, representing the noisy version of x after multiple noise addition steps. x_T encodes the residual information essential for reconstructing x during the reverse diffusion process. On the other hand, the multi-resolution feature z_i is generated by the stacked CAE, encapsulating multi-grained feature representations. We design the latent feature guidance diffusion model to generate hierarchical data samples from random noise conditioned on z_i.

In the forward diffusion process, a key observation is that the learning objective L in Eq. 5.9 relies on the marginal $q(x_t|x_0)$ instead of the joint $q(x_{1:T}|x_0)$. The forward process can be derived from Bayes' rule [193]:

$$q_\sigma(x_t|x_{t-1}, x_0) = \frac{q_\sigma(x_{t-1}|x_t, x_0)q_\sigma(x_t|x_0)}{q_\sigma(x_{t-1}|x_0)} \tag{5.11}$$

Indeed, each x_t in the generation process not only depends on x_{t-1} but also on x_0, rendering the process non-Markovian. Consequently, the term L in Equation 5.9 is not contingent on a specific forward procedure. Therefore, during the generation process, if $q(x_t|x_0)$ remains fixed, we have the flexibility to sample only a portion of the forward steps. This approach allows us to reduce the number of steps T and enhance data generation efficiency.

To guarantee $q(x_t|x_{t-1})$ satisfies distribution Eq. 5.4, $q_\sigma(x_{t-1}|x_t, x_0)$ should satisfy the Gaussian function [193]:

$$q_\sigma(x_{t-1}|x_t, x_0) = \mathcal{N}(\mu, \sigma_t^2 I). \tag{5.12}$$

$$x_0 = \frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta(x_t|z_i, t)}{\sqrt{\alpha_t}} \tag{5.13}$$

$$\mu = \sqrt{\alpha_{t-1}}x_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta(x_t|z_i, t) \tag{5.14}$$

Thus, given a subset $\{x_{\tau_1}, \cdots, x_{x_{\tau_S}}\}$ where τ is a sub-sequence of $[1, \cdots, t, \cdots, T]$ of length S, the model can be trained with arbitrary forward steps. We follow the setup to variance σ in [193] as follows:

$$\sigma_{\tau_i} = \sqrt{(1-\alpha_{\tau_{i-1}})(1-\alpha_{\tau_i})}\sqrt{1-\alpha_{\tau_i}/\alpha_{\tau_{i-1}}} \tag{5.15}$$

In the reverse diffusion process, Hippo removes the noise in data conditioned on the feature z_i. Consider the generative model is formalized as $P(x|z) = \frac{P(x,z)}{P(z)}$, and $P(z)$ is a deterministic function of x in the multi-resolution feature extraction model, so we only need to estimate $P(x, z)$. Thus, we propose to project the z_i to the existing diffusion model latent x_T:

$$\epsilon_\theta(x_T|z_i, T) = \epsilon_\theta(x_T|\emptyset, T) + s \cdot (\epsilon_\theta(x_T|z_i, T) - \epsilon_\theta(x_T|\emptyset, T)), \tag{5.16}$$

where s is the guidance scale, which is empirically set as 7.5 in our experiments. In Eq. 5.16, we can simultaneously train an unconditional diffusion model $p_\theta(x_T)$ and a conditional diffusion model $p_\theta(x_T|z_i)$. For the unconditional model part, we replace the z_i with a null vector with a probability such as 0.3 in our experiments. For the conditional model part, the denoising process is conditioned on z_i. Thus, we can guide the data generation toward the z_i during the training process.

It has been proven that the training objective in Eq. 5.9 is still applicable in the reverse process [193]. Finally, we train the diffusion model to approximate the conditioned probability distribution in the reverse process. Given the initial input noise $\epsilon_t \sim \mathcal{N}(0, I)$, we sample $x_{t-1} \sim p_\theta(x_{t-1}|x_t)$ as follows:

$$x_{t-1} = \mu + \sigma_t \epsilon_t \tag{5.17}$$

where μ is in Eq. 5.14. With the same model for predicted noise ϵ_θ, we can choose different σ_t without retraining the model to generate different samples. In the sampling phase, the reverse process in the DM can generate multiple data corresponding to a given latent feature representation z_i. Particularly, when $\sigma_t \neq 0$, the reverse process is non-deterministic. Larger values of σ_t introduce higher stochasticity, resulting in more variations of generated data guided by the same latent representation z_i.

5.4: Evaluation and Discussion

5.4.1: Experimental Setup

We implement the latent feature guidance diffusion model of Hippo using PyTorch [162]. The model is trained on a TensorEX Ubuntu 20.04 Server equipped with 256GB DDR4 memory, Intel(R) Xeon(R) Gold 5218R 2.10GHz CPUs, and RTX A6000 GPUs. During the training of the diffusion model, we randomly set the number of timesteps from 100 to 1000, as the model is designed to be trained with any number of forward steps, as described in Section 5.3.4. We conduct training for 100 epochs. In the phase of multi-grained data generation, we empirically set the inference time steps to 100. The guidance scale for hierarchical features is set to 2. For sensitive attribute and activity recognition, we employ 5-fold cross-domain validation to mitigate overfitting and ensure more consistent and fair evaluations. This approach also considers the size of the training and testing datasets.

5.4.2: Datasets

We evaluate Hippo on four public datasets: HARBox dataset [161], UCI HAR dataset [8], MotionSense dataset [142], and the daily and sports activity (DSA) dataset [14].

Dataset 1. The HARBox [161] dataset includes 9-axis IMU sensor data collected from 121 participants (17-55 years old) with 77 smartphone models in a crowdsourcing manner. The five activities of daily life are walking, hopping, calling, waving, and typing. The collected IMU data is resampled at 50 Hz, with a sliding window of 2 seconds. Thus, each data sample contains a 900-dimensional feature. Considering the large variety of users and devices, the dataset is heterogeneous enough to evaluate the robustness of different models.

Dataset 2. The MotionSense [142] dataset includes time-series data generated by accelerometers and gyroscopes in iPhone 6s, which is put into the front pocket of each participant. There are 24 participants, each with a unique combination of height, weight, age, and gender. They complete 6 activities in 15 trials: walking, jogging, sitting, standing, and downstairs and upstairs. The data is collected at a 50Hz sampling rate. The sensing data is collected by the SensingKit [104] from the Core Motion framework [47]. Note that to be consistent with the baselines, we use the four activities for comparison: downstairs, upstairs, walking, and jogging.

Dataset 3. The UCI Human Activity Recognition [8] dataset includes 30 participants performing activities while carrying a waist-mounted Samsung Galaxy S II. There are six daily living activities: walking, upstairs, downstairs, sitting, standing, and laying. The accelerometer and gyroscope data are collected at a 50Hz sampling rate. The sensing data is pre-processed by applying noise filters and segmented with a sliding window of 2.56 seconds and 50% overlap.

Dataset 4. The daily and sports activity [14] includes 19 types of daily and sports activities. Each of the 19 activities is performed for 5 minutes by every four females and four males. The 19 activities are walking, exercising on a stepper, exercising on a cross-trainer, rowing, jumping, playing basketball, and so on. The participants perform the activities in a flat outdoor area. The sampling rate is at 25Hz, and the 5-minute signals are divided into 5-second segments.

Table 5.1: Metadata information perturbation methods comparison using activity recognition and gender recognition models on the MotionSense dataset.

Method	Average Classification Accuracy on 5-fold Cross-Validation (%)								Acuracy Overall	
	Downstairs		Upstairs		Walking		Jogging			
	Activity	Gender	Activity	Gender	Activity	Gender	Activity	Gender	Activity	Gender
Raw Data	**95.6**	87.7	**95.4**	90.9	**98.5**	95.1	**97.3**	95.6	**96.7**	92.3
Differential Privacy [62]	82.5	78.3	83.3	79.1	84.6	80.2	84.2	79.8	83.7	79.4
AAE [141]	90.7	59.8	93.2	57.6	91.8	52.9	93.7	52.5	92.4	55.7
TIPRDC [115]	91.5	77.4	92.8	78.6	97.5	79.4	95.2	78.8	94.2	78.6
ObscureNet [84]	87.5	72.5	92.8	81.3	94.7	83.2	96.9	89.5	92.9	81.6
InfoCensor [252]	91.1	76.8	92.4	77.1	96.6	78.2	94.3	77.3	93.6	77.4
Hippo-Granu.0	92.4	54.2	92.8	54.8	97.6	**51.6**	96.5	**50.7**	94.8	52.8
Hippo-Granu.1	83.8	52.1	84.2	53.5	86.3	53.5	84.2	51.8	84.6	52.7
Hippo-Granu.2	66.2	53.5	65.7	52.8	63.8	54.7	67.2	51.8	65.7	53.2
Hippo-Granu.3	53.8	**51.8**	55.4	**52.1**	54.7	53.5	53.6	52.5	54.4	**52.5**

5.4.3: Metadata-level Information Evaluation

In this section, we evaluate the metadata-level overprivileged information protection. The metadata (*i.e.*, personal attributes) is auxiliary side-channel information in activity sensing data. We consider a scenario where an app requires the most fine-grained activity features, and the side-channel information is unnecessary but sensitive. The data utility is related to the find-grained features in raw sensor data and personal attributes are overprivileged information. In this case, Hippo only perturbs the attribute information while keeping the fine-grained activity features.

Baseline Methods

For metadata-level privacy protection, we compare Glint with five methods that garble personal attributes. We use the source codes provided by related authors to implement experiments. For semantic-level privacy protection, the existing methods are coarse-grained because they require users to provide private activity patterns, and then brutely remove or replace the segment of data that contains sensitive activity information. To the best of our knowledge, there is no existing work that considers fine-grained activity semantic-level privacy protection.

Baseline 1. The DP [62] method injects Laplace noise with various privacy budgets $\{0.1, 0.3, 0.7, 0.9\}$, and we report the average accuracy under the four parameters. The noisy data is used for activity recognition.

Baseline 2. The Anonymisation Autoencoder (AAE) [141] is a deep autoencoder model with a loss function to minimize the user-identify information and preserve the original activity information. The synthesized data from the decoder is used for activity recognition.

Baseline 3. The TIPRDC [115] is a framework to hide private information by adversarial training while retaining the original information by using a neural network-based mutual information estimator. The sanitized feature representations are used for activity recognition.

Baseline 4. The ObscureNet [84] is an encoder-decoder model to conceal private attributes in the time-series data. It learns latent features invariant to specific private attributes based on adversarial information factorization. The synthesized data is used for activity recognition.

Baseline 5. The InfoCensor [252] is also an information-theoretic framework to minimize the mutual information between the representations and the attribute. The feature representations are randomized by the parameterized Gaussian mechanisms and used for activity recognition.

Table 5.1 presents the performance of various models for attribute protection, focusing on the gender attribute. These models are trained to sanitize the gender attribute, utilizing two convolutional neural network models on the MotionSense dataset with 5-fold cross-validation for gender and activity recognition. When employing noise-based differential privacy (DP), increasing the strength of the added noise to better preserve sensitive information significantly compromises data utility. For instance, increasing noise to the extent that gender recognition accuracy decreases from 92.3% to 79.4% also results in a drop in activity recognition accuracy from 96.7% to 83.7%. In contrast, Glint aims to destroy sensitive attributes in the reconstructed data (*Granu.0*) while retaining as much of the original activity semantic information as possible. For instance, gender recognition accuracy on the *Granu.0* data is approximately 52.8%, akin to random guessing. However, activity recognition accuracy on *Granu.0* data is approximately 94.8%, comparable to model performance on raw data. Glint maintains high data utility owing to the learning-based denoising process during reverse diffusion.

Several methods, such as AAE, TIPRDC, ObscureNet, and InfoCensor, utilize information minimization in adversarial training. For instance, ObscureNet alters private attributes in latent features before synthesizing new data. Consequently, it can deceive gender inference models to an accuracy of approximately 20%. However, for a binary classification task, reversing the output is straightforward for a knowledgeable attacker familiar with ObscureNet's mechanism. Therefore, we report the reversed accuracy in Table 5.1. For instance, with the ObscureNet method, gender inference accuracy can be reversed to achieve approximately 80% accuracy. In contrast, Glint reduces the gender inference probability to around 50%, leading to the highest entropy (uncertainty) for attackers. Moreover, we evaluate gender information in multi-grained data. Multi-grained data generation is based on random noise conditioned on multi-resolution feature representations. As indicated in Table 5.1, gender information is perturbed in the multi-grained data, resulting in gender recognition becoming a random guess. In summary, for metadata-level overprivileged information protection, compared to the existing methods, Hippo preserves better data utility while perturbing overprivileged attributes.

5.4.4: Matadata-level Perturbation Analysis

In our scenario, suppose we have a dataset of sensing data from multiple users, denoted as X. Without loss of generality, each user has one segment of data x. Privacy is related to personal attribute information. Recall the definition of the differential privacy (DP) mechanism [13, 62]. Suppose we have two neighboring datasets X and X', where X differs from X' only by one person. The perturbation mechanism M satisfies (ϵ, δ)-DP if it holds:

$$Pr[M(x) \in O] \leq e^{\epsilon} \cdot Pr[M(x`) \in O] + \delta, \tag{5.18}$$

where $x' \in X'$ is the perturbed version of $x \in X$ via local perturbation M, so x and x' are neighboring inputs. O is any subset of outputs, ϵ is a metric of privacy loss (Note that ϵ in DP is different from $\epsilon(x_t, t)$ in diffusion models). The higher the ϵ, the larger the privacy loss. δ is the probability of violating the DP constraint.

The local perturbation mechanism M is defined as $M(x) = x + Z$, where Z is the Gaussian noise. Suppose we have a query function $f : \mathcal{X} \rightarrow \mathcal{R}^d$, which can be a classification model.

We define the sensitivity of the query function f as $s = max(||f(x) - f(x`)||_1)$, which is the maximum impact that one individual can have on the result of a query [225]. For example, for the gender attribute, f is a gender classification model. s is set as 1 considering the maximum difference of model prediction probability.

For the metadata-level perturbation, Z is the residual noise in the forward diffusion and reverse diffusion process. Recall that in the forward diffusion steps, Hippo is adding noise to x gradually as in Eq. 5.4 and Eq. 5.5. Suppose $M_t(r) = r_t + \epsilon(x_t, t)$, where $r_t = \sqrt{\frac{\bar{\alpha}}{1-\bar{\alpha}_t}} x_0$ and $\epsilon(x_t, t)$ is the merged multiple Gaussians $\mathcal{N}(0, \sum_1^T \sigma_t^2)$ given T steps. For each forward step t, we add the independent Gaussian noise $\mathcal{N}(0, \sigma_t^2)$ to the input of each step indexed by t. We can choose the parameters of additive Gaussian noise. If the added noise satisfies $\sigma_t = s\sqrt{2log(1.25/\delta)}/\epsilon$ in the forward diffusion step, then each step in the forward diffusion process satisfies (ϵ_t, δ_t)-DP. For example, in the experiments, we set the initial noise σ in the first step as 11.5. DP satisfies the composability property [62]. Thus, given a sequential composition of adding noise steps T, the $M_t(r)$ still satisfies the differential privacy. In the final forward diffusion step, the composition $M_t(r)$ is $(\sum_{t=1}^T \epsilon_t, \sum_{t=1}^T \delta_t)$-differentially private. Recall that $\sqrt{1 - \bar{\alpha}_t}$ in Eq. 5.5 is constant, and DP is robust to the post-processing [62]. Then the output $x_T = \sqrt{1 - \bar{\alpha}_t} M_t(r)$ in the forward diffusion process still satisfies differential privacy.

In the reverse diffusion process, Hippo learns to denoise by predicting the added noise in the forward steps with a learning function $\mathcal{F}$ following Eq. 4.13. The input of $\mathcal{F}$ is the final result x_T in the forward process. The output of $\mathcal{F}$ is x'. x' retains original semantic features as in x while metadata information is perturbed.

$$x' = \mathcal{F}(\sqrt{1 - \bar{\alpha}_t} M_t(r)). \tag{5.19}$$

Considering DP is robust to post-processing of $\mathcal{F}$, the whole diffusion process in Hippo is still differentially private. In Section 5.4.3, we have empirically shown the effectiveness of Hippo for personal attribute perturbation and activity feature preservation. However, due to the complex

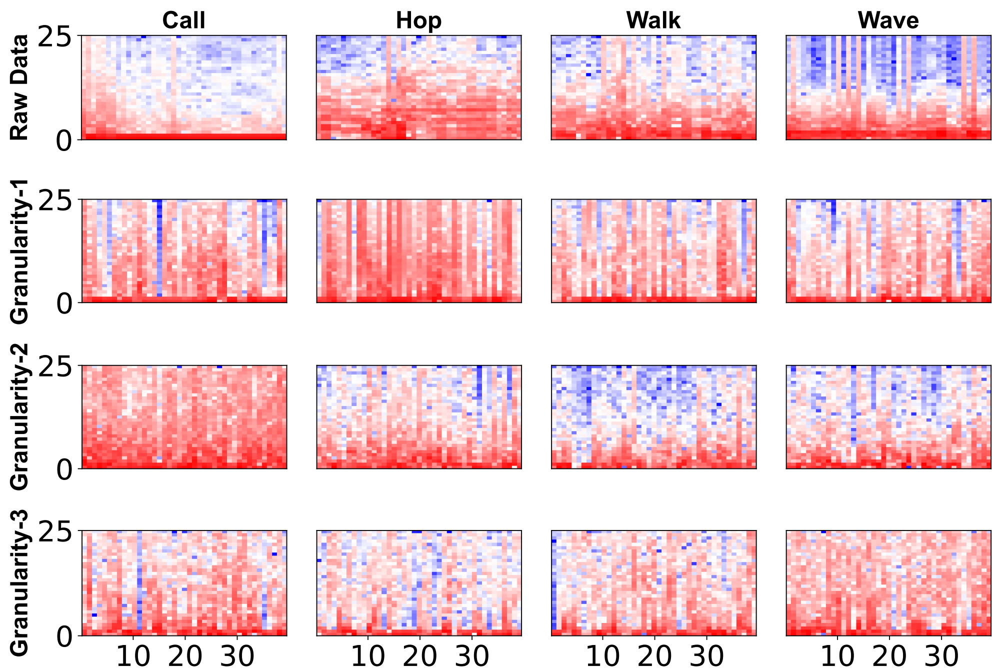


Figure 5.4: Spectrograms of multi-grained data for the accelerometer w.r.t. four activities. Red means high energy, blue means low energy, and white is in the middle. The x-axis is time (second), and the y-axis is the frequency (Hz).

deep learning mechanism in the reverse denoising diffusion process, obtaining the overall privacy guarantee of ϵ and δ in the whole diffusion process is open for future research.

5.4.5: Feature-level Information Evaluation

In this section, we address apps encountering feature-level overprivileged issues. To mitigate this, Hippo generates multi-grained data for reducing activity feature-level overprivileged information. This entails utilizing different resolutions of features for multi-grained data reconstruction. Consequently, apps may experience reduced accuracy when recognizing fine-grained activity semantics using coarse-grained data.

Hierarchical Data Visualization

While it is challenging to interpret activity semantics from time series graphs visually, we can gain insight into the hierarchical sensor data generated by Hippo by examining the different granularities of data in the frequency domain, as depicted in Figure 5.4. Specifically, we showcase four armintensive activities: calling, hopping, walking, and waving, sourced from the HARBox Dataset. (i) “Calling” involves a volunteer holding the device in a calling motion. (ii) “Hopping” entails a volunteer holding the phone and jumping on one foot. (iii) “Walking” occurs when a volunteer holds the phone and walks. (iv) “Waving” involves a volunteer holding the phone in hand and waving it. These visualizations offer insights into the frequency features of the sensor data for each activity, allowing for a deeper understanding of the hierarchical structure of the reconstructed data.

The multi-grained data is generated based on the guidance of different resolutions of feature representations extracted from raw data. Figure 5.4 provides visual representations of the spectrograms of multi-grained data generated by Hippo. In raw data, signal strengths vary across different frequencies. Conversely, in multi-grained data, signal strengths stabilize and become similar across various frequencies, indicating that detailed features are filtered out in coarse-grained reconstructed data. For instance, as depicted in Figure 5.4, the fine-grained features (in blue) of raw waving data gradually generalize across different granularities of data. Notably, the granularity-3 data of waving becomes unrecognizable. This is because detailed features are blurred in the coarse-grained (granularity-3) data. Similarly, the granularity-3 data of walking and calling also become unrecognizable. Therefore, if calling behavior is considered private, the user can opt to release granularity-3 data of calling to make the private mobile sensing data indistinguishable from other motion data. Overall, the visualization illustrates that Hippo can generate multi-grained mobile sensing data containing varying levels of feature information.

Clustering

We implement the K-means clustering based on feature embeddings learned via contrastive learning. With contrastive learning, we can learn a feature representation space where samples with different labels spread out and samples with the same labels remain close together. We use the following loss function of contrastive learning:

$$\mathcal{L}_{\mathrm{c}}(\mathbf{x}_i, \mathbf{x}_j, \theta) = \mathbb{1}[y_i = y_j] \| f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j) \|_2^2 + \mathbb{1}[y_i \neq y_j] \max(0, \epsilon - \| f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j) \|_2)^2, \tag{5.20}$$

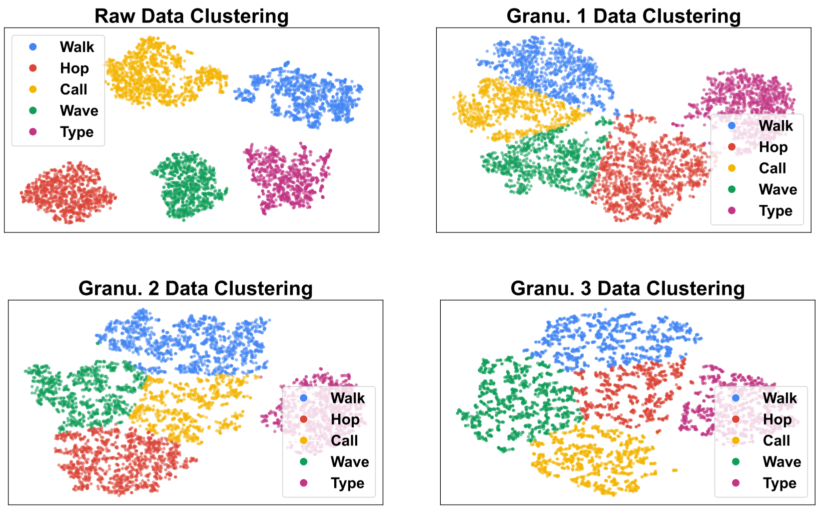


Figure 5.5: K-means clustering on 5000 randomly sampled HARBOX raw data and granularity 1-3 data with TSNE dimension reduction.

where $\mathbb{1}[y_i = y_j]$ means the value is 1 if two samples have the same label, otherwise is 0. $\mathbb{1}[y_i \neq y_j]$ means the value is 1 if two samples have different labels. ϵ is the upper bound distance to avoid the exceptional contribution of some dissimilar pairs.

Given the raw data from the HARBox dataset, we evaluate different granularity of data generated by Hippo. First, we train the Hippo to obtain a stacked CAE and a diffusion model-based data generator. Second, we extract different resolutions of feature representations from the raw data via the stacked CAE. For each resolution of feature representation, we calculate the mean of the representations Z_i of the raw data with the same label. In this way, we can mitigate the differences between different data samples of the same class. Third, we use the well-trained Hippo to generate new data samples from noise conditioned on Z_i. For example, the granularity-1 data is generated with the guidance of the layer-1 feature of the stacked CAE. We generate 5,000 samples for each level of granularity of data. Fourth, we train a contrastive model to obtain the contrastive feature representations. We use the t-SNE [206] method to implement dimensional reduction, and the K-means clustering on the feature representations of raw data, granularity-1 (Granu.1), granularity-2 (Granu.2), and granularity-3 (Granu.3) data.

We show that the reconstructed multi-grained data contains different granularity of feature information. Figure 5.5 illustrates the generalization of activity semantics. For instance, K-means can easily cluster different classes of raw sensor data. This is because raw data contains most fine-grained feature information. However, distinguishable fine-grained features are removed from coarse-grained reconstructed data. For Granu.1 data, the boundary of different activities becomes vague. For example, hopping and waving are hard to differentiate. The reason is that Hippo generates the new data with the guidance of the layer 1 feature of the stacked CAE. The layer 1 feature hides part of the fine-grained information of the raw data. Thus, though we obtain the feature representations via the contrastive learning model, the intrinsic boundary among different activities of data becomes blurry. For Granu.3 data, the representations of walking, hopping, calling, and waving are aggregated in a big cluster. Overall, we can see that Hippo can filter out fine-grained activity features and generalize the activity feature information in reconstructed data.

Classification

In this section, we implement classification models to evaluate the feature-level information in reconstructed data. The changes in classification performance show that the Hippo can generate multi-grained data that contains different granularity of activity features.

We compare Hippo with other generative models: Convolutional Autoencoder (CAE) [209], Generative Adversarial Network (GAN) [77]. Based on our designed multi-grained data generation framework including multi-resolution feature extraction and latent feature-guided data generation, we design CAE and GAN to generate new data. (i) For CAE, we extract the i-th layer feature z_i of raw data X with the encoder of CAE. Then, we map the z_i with a matrix W to the data X': $X' = Wz_i$. We control the error rate between X and X': $Err = ||X' - X||_2$ to retain the amount of information in X'.

(ii) GAN is based on adversarial training to learn to generate new data. We design GAN to implement the adversarial information factorization based on the specified layer of latent representations. Generally, the generative models such as GANs can be regarded as a nonlinear parametric mapping $g : Z \rightarrow X$ to the data manifold, Z is an open subset in $\mathcal{R}^d$, and X is an open subset in $\mathcal{R}^D$. The mapping is from a low dimensional space Z to a manifold M embedded in the higher dimensional space X. We propose that M approximates the different granularity of the data manifold. We study the manifold approximation error of the generator, which minimizes a chosen divergence measure between the raw data distribution and the distribution of different granularity of the sensor data.

Ideally, we can learn the latent representation $z = h(x)$, where $z \in Z$, so that $g(h(x)) = x$. However, in practice, g can only learn an approximation to the true data manifold, which causes the approximation error. We utilize the approximation error to generate the privacy-preserving item. We control the semantic closeness between x and $g(h(x))$, so the class label has different semantic granularity. The tangent space of a manifold generalizes to higher dimensions. We use the $J_x h$ to estimate the tangent space, of which the rows of $J_x h$ are the directions that approximately span the tangent space to the data manifold at x.

Note that several places can cause approximation errors, $M \sim$ Manifold, $g(h(x)) \sim x$, $h(g(z) \sim z$, so we only consider dominant tangent directions in the row span of $J_x h$. These can be obtained by the SVD on the matrix $J_x h$, and get the right singular vectors. However, this process is expensive because, for each data sample, SVD needs to do it one by one. Kumar*et al.* [112] propose to use GAN to estimate the tangent space to the data manifold. In this chapter, we take the stacked autoencoder, with the encoder-generator-discriminator triple to get the dominant tangent direction. A distribution p_z over the space Z results in a distribution p_g over the space X, and we can get the sample $x = g(z)$ from the distribution by sampling $z \sim p_z$. The discriminator approximates a divergence measure between p_g and the real data distribution p_x. We use $KL[p||q]$ to denote the KL divergence between the distributions p and q as follows:

$$\begin{aligned} \Delta_{KL}(r, x, \theta) &\equiv KL[p(y|x, \theta)||p(y|x + r, \theta)] \\ r_{adv} &\equiv \arg\max_r \{\Delta_{KL}(r, x, \theta); ||r||_2 \leq \varepsilon\} \end{aligned} \tag{5.21}$$

We train the generator using feature matching where the generator minimizes the mean discrepancy between the features of raw data and generated data from an intermediate layer l of the discriminator f. Meanwhile, we maximize the mean discrepancy between the features of raw

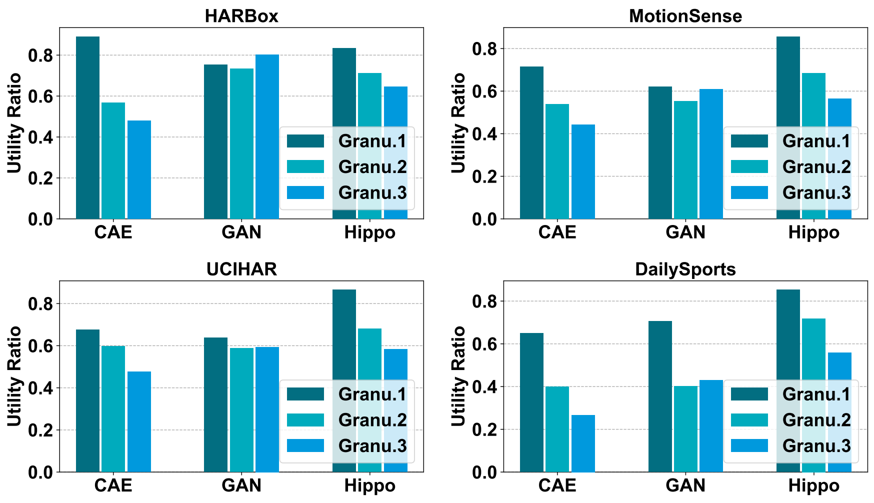


Figure 5.6: Comparison of different generative models.

data and generated data from the layer $l + 1$ of the discriminator f as follows:

$$L_g = ||E_x[f_l(x)] - E_z[f_l(g(z))]||_2^2 - ||E_x[f_{l+1}(x)] - E_z[f_{l+1}(g(z)]||_2^2 \tag{5.22}$$

We employ the same CNN model structure, comprising two convolution layers and two fully connected layers, to evaluate datasets generated by different generative models. Specifically, for CAE data evaluation, we train the CNN model on datasets generated by CAE. We set three thresholds 0.001, 0.01, 0.1 to generate three levels of granularity of data with CAE. Similarly, for GAN, we employ adversarial training to train the generator conditioned on layer 1-3 features. We then train Hippo on these datasets and generate new data with the guidance of layer 1-3 features. Figure 5.6 illustrates the differences in utility across different granularities of data. Utility is computed using the activity recognition utility function defined in Section 5.2.2. Hippo successfully generates multi-grained data with varying amounts of semantic information. For example, on the MotionSense dataset, Hippo achieves the highest utility score of 0.856 for the Granu.1 reconstructed data. Granu.1 data guided by the layer-1 latent features preserves the most activity information from the raw data. The utility scores for Granu.2 and Granu.3 data are 0.685 and 0.564, respectively. The decrease in utility score indicates that Hippo can generate multi-grained data with hierarchical semantics. By filtering out fine-grained activity features during the data reconstruction process, Hippo makes fine-grained activity semantics indistinguishable.

Hippo surpasses existing generative models in several aspects. Firstly, while the CAE method can control the information in the generated data, setting the E_{rr} parameter to control data granularity is challenging due to the heterogeneity of sensor data. Different sensors or activities may exhibit varying magnitudes of data, complicating the task. For instance, the utility scores of Granu.1 data on HARBox, MotionSense, UCIHAR, and DailySports using CAE are 0.89, 0.715, 0.676, and 0.65, respectively. In contrast, for Hippo, the utility scores of Granu.1 data on these datasets are 0.834, 0.856, 0.864, and 0.852. The utility scores of CAE on the same granularity exhibit higher divergence than those of Hippo. Secondly, training GAN to generate high-quality data consistently in practice is challenging. While GANs are designed to be conditioned on latent features, controlling the amount of information contained in the generated data is difficult. For instance, the CNN model trained on Granu.2 data may perform better than the model trained on Granu.1 data. Additionally, samples generated by GANs often exhibit more noise, resulting in a more diverse utility compared to other methods, as shown in Figure 5.6. In contrast, our designed model Hippo can consistently generate high-quality data.

5.4.6: Feature-level Data Granularity Analysis

Mobile activity sensing data often contains fine-grained action feature information beyond the requirements for user-intended purposes. We consider a scenario where an app requires only a certain level l_k of activity semantics. In this case, Hippo will generate new data X_r from random noise conditioned on a certain resolution feature. The fine-grained features are removed in X_r. As a result, the X_r can be predicted into a coarse-grained activity semantics, which is a category of ℓ distinct fine-grained semantics.

On one hand, Hippo generates new data x' from random noise conditioned on a specific resolution feature z as in Eq. 5.16. Therefore, the feature of x' depends on the z. From Lemma 1, we know that the higher dimensions of z in low-level CAEs encode finer-grained data features, and the lower dimensions of z in high-level CAEs preserve coarse-grained semantic information [172].

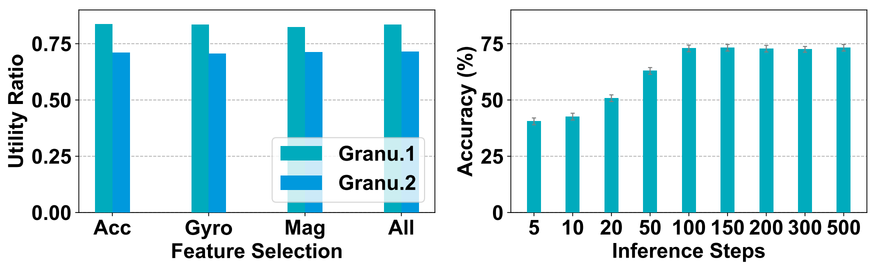


Figure 5.7: The impact of feature selection and inference steps.

Suppose $x_i' = f(N; z_i)$ and $x_j' = f(N; z_j)$, where $i > j$ and z_i is in low-level CAE and z_j is in high-level CAE, then data x_i' has fine-grained features than x_j'. On the other hand, Hippo provides two ways to generate diverse data for each granularity. (i) Recall from Eq. 5.17, we can choose different σ to generate multiple data conditioned on a given feature representation. (ii) Various noise levels related to T correspond to different scales of features captured by the diffusion model: the greater the noise level, the larger the scale of features [90]. Thus, Hippo can learn different scales of data features and generate multi-grained data according to the latent feature z and diffusion noise level.

5.4.7: Ablation Study and Efficiency Analysis

The impact of input feature

Our initial assessment focuses on how feature selection influences the effectiveness of activity recognition, utilizing the HARBox Dataset for this analysis. Figure 5.7 elucidates that 'Acc' represents a 3-axis accelerometer, 'Gyro' denotes gyroscope modality, 'Mag' signifies magnetometer modality, and 'All' encapsulates all 9-axis modalities. We calculate the utility ratio as outlined in Section 5.2.2, defined by the quotient of model accuracy on reconstructed data to that on original data. Observations from Figure 5.7 reveal that the utility derived from reconstructed data using Hippo maintains consistency irrespective of feature (modality) selection. For example, the utility ratios for Granu.1 reconstructed data across the four modalities are 0.836, 0.833, 0.823, and 0.834, respectively. This uniformity in utility ratios across various feature choices further validates the resilience of Hippo towards feature selection.

In addition, we recognize that the sequence in which input features are presented and their sampling frequency can influence the accuracy of activity recognition. However, it is fundamentally presumed that the order of features should not significantly alter the effectiveness of feature utilization for recognizing activities. Consequently, the feature extraction capability of Hippo remains unaffected by the sequencing of input features. Specifically, convolutional layers demonstrate resilience against variations in inputs (e.g., through image augmentation techniques [186]) during feature extraction. In a similar vein, Hippo exhibits stability when dealing with data sampled at varying rates. For example, while the DSA Dataset [14] operates at a 25Hz sampling rate, the MotionSense dataset [142] utilizes a 50Hz rate. Figure 5.6 illustrates that Hippo maintains consistent performance across different sampling frequencies. In essence, provided that the activity recognition models are adaptable to feature variations, Hippo is also robust to such changes will persist.

The impact of inference steps

The influence of inference steps within the diffusion model on data generation is assessed utilizing the HARBox Dataset. We document the accuracy of activity recognition using Granu.1 data generated by Hippo at various inference stages. As depicted in Figure 5.7, with a minimal inference step count, such as 5, the accuracy of activity recognition dips to 40.5%, indicating suboptimal data quality. However, as the inference steps are incremented to 100, there is a noticeable enhancement in accuracy. Beyond 100 steps, the accuracy levels off, suggesting a plateau in quality improvement with additional steps. This observation underscores that a higher number of inference steps does not inherently equate to superior data quality.

Runtime Efficiency

Given a 50Hz sampling rate in the sensing application and a 1-second data buffering window for reconstruction, we find that on a server equipped with NVIDIA RTX A6000 graphics cards, Hippo is able to reconstruct a new data version from a sequence of size 100×9 within 0.61 seconds using 100 inference steps in the diffusion model (DM). This efficiency ensures that Hippo can generate new data within the allocated 1-second buffer time. Furthermore, the storage footprint of Hippo is relatively minimal, with its model size being just 49.5 MB when saved to disk. This compact size, combined with advancements in hardware technology, suggests the potential for deploying Hippo on mobile devices. For instance, the iPhone 14, with its 6 GB RAM and A15 Bionic chip featuring a 5-core GPU and 16-core Neural Engine, already supports complex operations such as text-to-image generation [158]. An alternative approach could involve running Hippo on a trusted edge server, allowing various applications to access and utilize the processed sensing data. Moving forward, our efforts will focus on enhancing Hippo's computational efficiency through techniques such as quantization [167], model pruning [259], and model parallelism [185], to further facilitate its adoption in diverse computing environments.

5.4.8: Case Study

Pedometer Application

The pedometer, a common tool in our daily routines, allows individuals to track their daily steps using smartphones, smartwatches, or smart bracelets. For the sake of this analysis, assume the user engages in four primary activities throughout the day: walking, calling, hopping, and waving. Among these, the user wishes to keep the calling activity private, as it represents sensitive information. However, there is a potential privacy concern: if the pedometer service provider excessively gathers mobile sensing data, it could inadvertently expose sensitive information, such as instances when the user is making calls during class. In this scenario, an attacker's primary objective would be to discern this sensitive activity, specifically identifying when the user is calling during class time, thereby breaching the user's privacy. This highlights the importance of developing mechanisms to safeguard sensitive information while still utilizing the benefits of mobile sensing technologies.

To safeguard sensitive activity information while maintaining the utility of step counting, users can employ Hippo to reconstruct motion sensor data. Consider four primary activity time windows, denoted as X_1, X_2, X_3, X_4, corresponding to walking, calling, hopping, and waving, respectively.

The parameter Δ in Eq. 5.2 is crucial as it governs the balance between reducing overprivileged information and retaining essential features for step counting in the reconstructed data. Importantly, the setting of Δ needs to be personalized based on the user's profile and the nature of the activities. For instance, in the context of X_2 (calling), if an attacker is able to discern the activity's semantic meaning with a certain probability distribution (e.g., $0.9, 0.025, 0.05, 0.025$ for calling, walking, waving, hopping, respectively), the entropy $H(D)$ in Eq. 5.2 is calculated to be 0.186. By setting Δ to 0.25, we aim to limit the amount of sensitive information in the reconstructed data, ensuring that $H(G)$, the entropy after reconstruction, should exceed $H(D) + \Delta = 0.436$. Through this approach, Hippo can adaptively reconstruct the sensor data, applying a degree of feature guidance to maintain step-counting functionality while protecting sensitive activity information.

In a collaborative effort between the service provider and the user, the hyperparameter i can be established for the i-th level of feature guidance. If a user desires to retain the full utility of sensor data, they can opt for granularity $i = 0$, designated as Granu.0. This setting keeps the raw data X_i within a time window while merely altering metadata to protect privacy. For instance, to obfuscate sensitive information within data X_2, the user might choose granularity $i = 1$, termed Granu.1, which abstracts the specific details of the sensitive activities while ensuring that critical features for step counting, such as peaks and valleys in X_1, are preserved. Upon assessing the overprivileged information in the reconstructed Granu.1 data X_2', it is observed, as illustrated in Figure 5.4, that the Granu.1 data for calling and waving motions appear similar, indicating the removal of detailed features that could distinguish between these two activities. Consequently, an attacker's ability to identify the possible semantics of X_2' as calling, walking, waving, hopping shifts to probabilities $0.41, 0.15, 0.39, 0.05$, resulting in $H(G)$ being 0.507 and $H(G; D)$ being 0.321. Given that $H(G; D)$ surpasses the set threshold $\Delta = 0.25$, it confirms that the requirement for reducing overprivileged information is met. This approach effectively mitigates the risk of exposing sensitive details through feature-level data in mobile sensing, thereby enhancing user privacy while retaining essential functionality.

The effectiveness of reconstructed multi-grained data for step counting is assessed by collecting

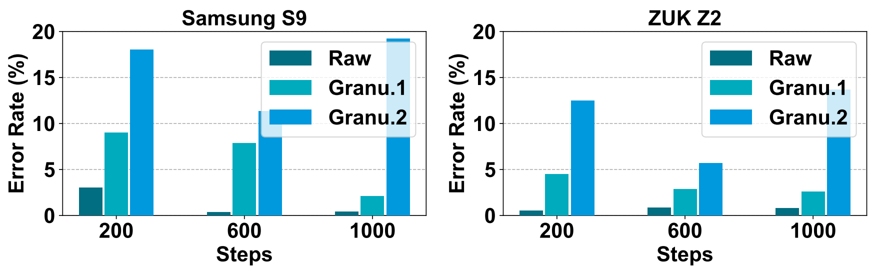


Figure 5.8: The pedometer performance on multi-grained data using different devices.

accelerometer and gyroscope data from a user through two Android smartphones, Lenovo ZUK Z2 and Samsung Galaxy S9. The user performs walking activities for 200, 600, and 1000 steps, with data captured at a 50 Hz sampling rate. Utilizing Hippo, we generate Granu.1 and Granu.2 data variants. Subsequently, the Pydometer library [46] is employed to determine if the reconstructed data maintains its utility for step counting. Figure 5.8 demonstrates that Granu.1 reconstructed walking data retains step count information effectively, exhibiting an error rate below 5% regardless of the step increment. This indicates that for pedometer services, which primarily rely on peak and valley information, a granularity setting of $i = 1$ ensures the preservation of essential utility in the reconstructed data. Consequently, we can reconstruct sensing data with level-1 feature guidance to mitigate the risk of sensitive "calling" information leakage, all while maintaining the functional integrity of pedometer applications.

Fall Detection Application

Mobile devices such as cStick [170], designed as a monitoring tool for fall detection in older adults, highlight the need for selective data usage in sensitive contexts. While cStick is focused on detecting falls, it inevitably collects a broader range of activity data, some of which might be sensitive and not directly relevant to its primary function. This broader data collection could lead to overprivileged access issues, where more information is gathered and potentially exposed than necessary for the device's intended purpose.

To address this, Hippo can be employed to reconstruct the sensing data, thereby perturbing sensitive information while maintaining crucial information related to fall detection. By selectively filtering the data through Hippo, the device can still perform its core function of fall detection

Table 5.2: Fall detection results on multi-grained data.

Ganularity	Raw Data	Granu. 0	Granu. 1	Granu. 2	Granu. 3
Utility	1	1	0.966	0.874	0.798

without retaining unnecessary details about the user's other activities. This approach ensures that the cStick continues to serve its purpose effectively, providing essential support and safety for older adults, while respecting their privacy and minimizing the risk of exposing sensitive information.

The generalization capability of Hippo is assessed by training the model using the HARBOX and MotionSense datasets, then applying the trained model to process the cStick dataset. A RandomForest classifier, trained with accelerometer readings from the open cStick dataset, is utilized to detect falls. The evaluation focuses on the utility of reconstructed multi-grained data. According to Table 5.2, when the RandomForest classifier is trained and tested with raw data, it achieves 100% accuracy in fall detection. Hippo, through its diffusion process, can alter sensitive attributes in raw data to generate Granu.0 data, which safeguards metadata-level sensitive information without compromising the utility for fall detection—retaining a utility value of 1, equivalent to that of the raw data.

When examining the Granu.1 data, generated with guidance from the first layer feature of the stacked CAE, there is a slight decrease in detection accuracy due to false negatives and positives, as detailed in Table 5.2. The accuracy for fall detection using Granu.1 data is 96.6%, indicating a utility of 0.966. This decrease in accuracy illustrates the trade-off involved in enhancing privacy by removing detailed motion features to produce more coarse-grained reconstructed data. In essence, Hippo allows users to balance the level of detail in the reconstructed data with their privacy needs. If high accuracy for specific activity recognition is crucial, users can opt to retain more detailed activity features in the reconstructed data. Our proposed multi-grained data reconstruction approach offers fine-grained control over information release, enabling users to tailor the balance between privacy protection and the utility of sensing data for specific applications such as fall detection.

5.4.9: Discussion

Data Granularity

In Hippo, the data granularity level is a critical hyperparameter that dictates the resolution of feature extraction, influenced by the number of Convolutional Autoencoders (CAEs) employed for multi-resolution analysis. It is important to note that data granularity and semantic granularity are not directly correlated; their relationship varies depending on the specific use case. However, there are overarching guidelines for using different granularity levels in Hippo: (i) Mitigating Metadata-level Overprivileged Issues: When the objective is to address concerns at the metadata level, users can opt for "Granularity-0" data reconstruction. This approach leverages Hippo to obscure personal attributes while preserving the activity features found in the raw data. Essentially, this level aims to protect the user's privacy without significantly impacting the utility of the data for its intended purpose, such as activity tracking or fall detection.

(ii) Protecting Feature-level Overprivileged Issues: If the concern extends to feature-level privacy, users can generate "Granularity-i" data. This is achieved by selecting the i-th layer of features from the stacked CAEs, where $i \geq 1$. Each incremental level of granularity (increasing i) typically offers a higher degree of data abstraction, removing more detailed information to enhance privacy. However, this also means potentially reducing the utility of the data for certain applications, as more fine-grained features may be filtered out. By adjusting the granularity level, users have the flexibility to find a balance between privacy protection and data utility, tailoring output from Glint to suit the specific requirements and sensitivity of their application.

The configuration of the granularity level i in Hippo is intricately linked to the specific utility requirements of applications and the need to mitigate overprivileged information exposure. Given the wide array of human activities and the varying contexts in which they occur, it is essential for users and service providers to collaboratively determine the most appropriate granularity level i. This process typically involves the following steps:

(i) Evaluation by Service Providers: Initially, service providers should assess the hyperparameters of Hippo, such as the number of stacked CAEs, using public datasets. This evaluation helps in understanding how different configurations impact the data reconstruction process and the resulting data granularity.

(ii) Generation of Multi-Grained Data: Based on the evaluation, service providers can then generate data at various levels of granularity using different numbers of CAEs. This produces a range of reconstructed datasets, each offering a different balance of privacy protection and utility.

(iii) Collaborative Selection: Users and service providers can collaboratively review these multi-grained data sets to identify which granularity level aligns best with the desired level of activity semantics and privacy protection. This empirical approach allows stakeholders to make informed decisions that cater to the specific needs of the application.

Additionally, the threshold Δ, as discussed in Section 5.2.3, plays a crucial role in determining the balance between privacy and utility. It is tailored to align with the specific activity recognition scores, as illustrated in the pedometer example in Section 5.4.8. Since setting the ideal Δ can be challenging without comprehensive context, involving a trustworthy third party might be necessary to assist users in defining this hyperparameter accurately.

It is worth noting that completely eliminating feature-level overprivileged information without compromising necessary activity features is a challenging endeavor. Hippo offers a pragmatic solution by allowing the selective release of multi-grained data, thus providing a mechanism for data minimization. Recognizing the complexities surrounding data minimization, future efforts will focus on enhancing Hippo's user-friendliness and effectiveness in real-world applications, ensuring it remains a viable tool for balancing data utility and privacy.

Implementation

Hippo serves as a middleware for reconstructing sensor data at multiple granularity levels, enabling the perturbation of metadata and the removal of detailed features to address privacy concerns. Integrating Hippo within a secure realm, such as ARM TrustZone, enhances data security by providing an isolated environment where raw mobile sensing data is exclusively accessed and processed by Hippo. This setup ensures that only preprocessed data, which has been perturbed or generalized according to the specified granularity level, is accessible to other applications.

In practice, Hippo's functionality allows for tailored data access: For applications that necessitate raw activity features, Hippo minimally perturbs only the metadata, ensuring that essential details for the application's functionality are preserved. For applications where only broad activity patterns are necessary, Hippo produces data at a coarser granularity, removing detailed features to enhance privacy. Moreover, Hippo's integration with privacy-check mechanisms such as MaskIt provides a dynamic framework for determining when data reconstruction should be applied, catering to the diverse privacy needs across different scenarios. This adaptive approach allows Hippo to offer a customizable privacy-preserving solution that aligns with varying requirements, thereby providing a more secure and flexible data-handling process within mobile and wearable device ecosystems.

As a learning-based method, Hippo can be generalized to new users and tasks by training on large-scale datasets considering the diversity of user activities. Hippo perturbs the metadata information without changing activity feature information. In addition, Hippo can change the activity feature granularity by generating multi-grained data, which can be applied to new scenarios or tasks. For instance, Hippo can be applied to defend against eavesdropping attacks that utilize motion sensors on smartphones [6, 12, 92, 149] by removing fine-grained features. Furthermore, in this chapter, we assume the apps are trusted to honestly claim the intended functionalities considering the law and policy regulations. In the future, we will consider how Hippo prevents apps from overclaiming the requirements for intended functionalities. Alternatively, we will enhance the mechanism of setting data granularity in Hippo to avoid the involvement of third-party services.

5.5: Summary

In this chapter, we address the challenges of metadata-level and feature-level overprivileged issues prevalent in mobile sensing applications. Drawing on the layered complexity of human activities and the capabilities of generative artificial intelligence, we introduce Hippo, a system designed to produce data at multiple granularity levels. This system enables the perturbation of extraneous metadata and the elimination of unnecessary activity details, offering a tailored approach to data privacy. For tackling metadata-level overprivileged issues, Hippo employs a diffusion process that obfuscates private attributes, establishing a theoretical linkage between the diffusion model and the principles of differential privacy. On the feature level, Hippo adeptly removes detailed activity information by generating data at varying granularities, allowing the same data to be interpreted at different levels of detail depending on the requirements of applications. Users gain the capability to control the granularity of the data they release through Hippo, effectively addressing overprivileged data access concerns. Our implementation includes a variety of data visualization, classification, and clustering tasks, along with two case studies across different datasets to assess the efficacy of Hippo in generating multi-grained data. The comprehensive evaluations demonstrate that Hippo is proficient in managing overprivileged issues in mobile activity sensing, offering a fine-grained approach to controlling data access and usage. This enables a balance between data utility and privacy, providing users and developers with a powerful tool to manage data privacy in mobile sensing applications.

CHAPTER 6: CONCLUSION

In this dissertation, we investigate the complex interactions between users, devices, and the environment. In particular, we explore the potential security and privacy issues behind the interactions. There are external attacks and user misconfiguration issues that could lead to unexpected and insecure IoT device interaction, causing damage to the physical world. To enforce the IoT device interaction security, we proposed the interactive threat detection system, Glint, to identify the potential interactive threats in smart home automation configuration rule data. Then, considering the privacy concerns of sharing smart home data for threat detection model training, we designed the federated training paradigm to collaboratively train the threat detection model with multiple local households. In addition, we present the Monte Carlo beam search-based method to discover the root causes of the interactive threat in the IoT device interaction graph.

We then pay attention to the privacy issues behind the interaction between users and mobile devices such as smartphones. In particular, we present a new acoustic-based method to recognize the facial expressions of users, which can be used to expose the emotional privacy of users in real time. Meanwhile, the proposed acoustic-based facial expression method can be used for digital content recommendation and mental health care. Furthermore, we point out the overprivileged issues in the mobile activity sensing data, where the sensing data contains more information than is required for application purposes. Considering that excess information may contain sensitive information of users, we propose Hippo to reconstruct multi-grained mobile activity sensing data. In this way, Hippo can empower users with the choices to perturb metadata information and obscure sensitive activity information that is unintended to be released by users.

6.1: Summary of Contributions

We addressed multiple challenges in developing data-centric AI techniques for interaction security and privacy in the Internet of Things. From the data perspective, we solve the data scarcity problem by fusing data from multiple sources and collecting data from volunteers in real-world scenarios. (i) We have crawled automation rule description data from public smart home control platforms and built a correlation classification model to find correlations between different rules. Then, we synthesize a large scale of interaction graph datasets by randomly composing different numbers of rules with correlations into graphs. (ii) We collect acoustic-based facial expression signals from 20 volunteers. The collected facial expressions are anger, happiness, disgust, sadness, fear, and surprise. To ensure diversity in facial expressions, the volunteers vary in skin color and come from various parts of the world, with their ages ranging between 20 and 38 years. The data collection process was conducted intermittently over the span of a week. The dataset amassed exceeds 1 GB when saved in plain text format. Finally, we obtained 20,535 samples using the sliding windows segmentation method, with each window being 0.25 seconds in length.

From the model perspective, we design different deep-learning models to learn features for different tasks. (i) We design a novel graph neural network model to learn interactive threat patterns in complex interaction graphs. Specifically, considering the threat pattern could exist at different scales in a heterogeneous interaction graph, the designed ITGNN model includes metapath-based node transformation, multi-scale graph generator, and multi-scale graph fusion. In addition, we propose to apply transfer learning techniques to enhance the model performance on different platforms with data imbalance issues. (ii) We propose a contrastive attention-based domain adaptation model to extract acoustic-based facial expression features and identify six basic facial expressions. Specifically, we incorporate testing data without labels into the training process by designing the pseudo-label generation process. The external attention learning method makes the neural network focus on important features that are related to differentiating different facial expressions. In addition, the external attention mechanism can force the model to learn correlations among different expression samples. (iii) We present a latent diffusion guidance data generation method based on a convolutional autoencoder and diffusion model. The designed method can perturb metadata information and generate multi-grained data that contains different granularity of activity semantic information.

From the system perspective, we design multiple systems to explore and defend against the potential security and privacy attacks in the IoT. (i) We design a graph learning-based interactive threat detection system, which can discover the potential new threat across close-source heterogeneous systems. (ii) We propose a layer-wise dynamic clustering-based federated learning system to train the graph learning-based threat detection model while preserving the privacy of smart home usage data. (iii) We design the acoustic-based facial expression system using the microphone and earpiece speakers in the commercial smartphone, which can be used to understand users' emotional status. (iv) We design a mobile seeing data reconstruction system, which can be used as a middleware between sensor manager in the operating system and various application. The designed system can perturb sensitive metadata information and obscure sensitive activity features that are unintended to be released by users.

6.2: Discussion of Limitations

While this dissertation has yielded significant insights, it is important to consider the following limitations that might influence the generalizability and applicability of the results. Addressing these limitations can provide a clearer path for future studies to build upon and refine the work presented here, ensuring that subsequent efforts are more robust and comprehensive.

Graph Learning for Threat Detection

This dissertation proposes a graph neural network (GNN) based solution for detecting interactive threats among smart home automation applications. natural language processing (NLP) techniques are employed to extract automation rule information from app descriptions so that the source code of applications is not required for the interaction analysis. However, we acknowledge there are limitations to implementing the proposed system.

First, we build the interaction graph dataset using NLP techniques, which discover semantic correlations between different rules that are crawled from various platforms. However, due to the heterogeneous IoT platforms, this dissertation acknowledges that different platforms need different techniques for analyzing IoT apps and extracting their app semantics. Moreover, the IoT data could have data missing and corruption issues. We need to pay more attention to detecting and fixing data corruption issues in the IoT environment. Nevertheless, our goal is to create an interaction graph dataset to train the graph learning-based threat detection model. The evaluation has shown that the created dataset from publicly available platforms can empower the threat detection model to achieve decent performance in real-world settings.

Second, for the interaction graph construction in a real environment, we mainly use the device names and status information from event logs to build the correlation between different rules, that are associated with different devices. In the threat detection process, we approach the interaction graph as a static entity, essentially viewing it as a snapshot within a dynamic setting. However, there are time delays when extracting event logs and building the interaction graphs. As a result, if two events occur synchronously or nearly at the same time, there could be delays or inaccuracies in building the interaction graph in real-time.

Third, the graph learning-based method is essentially a probabilistic model. The model generally requires a large amount of data to perform well. Insufficient data can lead to poor estimates of the probabilities and model parameters, thereby affecting the overall performance. In addition, the graph neural network models, involving complex interactions or numerous parameters, can be difficult to interpret. This complexity can make it challenging for users to understand the reasoning of models or to extract actionable insights. In the future, we aim to improve the interpretability of AI-based models. For instance, the saliency maps can highlight the parts of input data that are most important for the predictions of a model. The SHapley Additive exPlanations (SHAP) uses game theory to explain the output of any machine learning model. It assigns each feature an importance socre for a particular prediction, providing a more comprehensive understanding of model behavior. The layer-wise relevance propagation method decomposes the output decision to reveal the contribution of each input feature to the final decision, helping to visualize how individual features contribute to predictions. Implementing these techniques can help make machine learning models more understandable to both developers and users, ultimately leading to better trust and more effective deployment of AI technologies.

Fourth, to facilitate classification, we have manually labeled every interaction graph as “safe” or “unsafe”, which takes much effort. Given the thousands of possible graphs of apps and devices on different platforms, labeling a large number of graph data could be prone to error. Besides, in different scenarios, different users have different security requirements. What is typical in one house may be atypical in another. For example, opening a window at night may invite robbers. However, if this is a high-rise apartment, then it would not be a threat to certain users. In the future, we plan to enhance the precision of our data labeling and introduce more detailed threat-type labels with the assistance of experts and current threat identification tools. Furthermore, this dissertation introduces the use of transfer learning techniques, and we will further explore how to effectively balance the original local data on a platform with transferred models from other platforms.

Federated Threat Detection Model Training

In this dissertation, we design a federated learning (FL) approach on non-IID data to learn the threat patterns. However, FL faces several security and privacy challenges. (i) FL suffers from data poisoning attacks. Malicious participants can manipulate the training process by injecting corrupted data or gradients, which can lead to a compromised model that fails to perform accurately or behaves undesirably when triggered. (ii) FL is vulnerable to the inference attacks. Despite not sharing data directly, subtle information about the training data can still be inferred from the shared model updates. For instance, membership inference attacks can determine if a particular data point was used in the training dataset, potentially revealing sensitive information. Techniques such as differential privacy are often needed to mitigate this risk, but they can also degrade the model’s performance. (iii) FL is challenged by significant communication overheads and security concerns. FL requires frequent exchanges of information (model updates) between clients and the central server, which can expose the process to interception and tampering attacks. Secure communication protocols are essential to protect these exchanges. Addressing these issues is crucial for the successful deployment of FL in privacy-preserving threat detection in smart homes.

Acoustic-based Facial Expression Sensing

This dissertation designs an active acoustic facial expression recognition system that leverages the earpiece speaker and microphone at the top of the phone body. This dissertation proposes a contrastive external attention model to learn representative and robust facial expression features while eliminating the background noise. There are some factors and cases that need to be considered in future work to enhance the robustness and practicality in real life.

First, the relative position between the user and the smartphone is flexible. When a user is interacting with a smartphone, the position of their face in relation to the device can change significantly. For instance, the distance and angle between the user's face and the smartphone can vary widely. Additionally, users might turn their heads to engage in conversation with others, often not facing the smartphone directly. In addition, the distance between the user and the mobile phone has a relatively strict limit in this dissertation, namely 20-50cm, but there is often a small distance in real life. Therefore, in the future, we will consider more factors such as head orientation, and distance to evaluate the performance of FacER.

Second, smartphone designs differ significantly. For instance, the placement of the microphone and earpiece speaker varies across different brands. Consequently, in the process of removing direct path signals, which travel directly from the speaker to the microphone, the specific design parameters—particularly the direct path data from the speaker to the microphone—play a crucial role. In the future, we plan to expand our evaluation to include various smartphone models and design advanced algorithms to mitigate the negative effect of direct path signal noise.

Third, enhancing the continuous recognition of facial expressions requires additional effort. In this dissertation, accurate facial expression recognition necessitates the initial segmentation of acoustic signals that reflect facial expressions. However, determining the exact start and end of a facial expression poses a challenge. On one hand, the duration of the same facial expression can vary considerably when performed by the same individual. On the other hand, the signal variations stemming from facial expressions are quite subtle. We have collected data for each facial expression over five seconds. In the future, we aim to explore solutions for segmenting acoustic signals to effectively extract facial expressions, thereby facilitating real-time facial expression recognition.

Multi-grained Sensing Data Reconstruction

This dissertation provides a learning-based data sanitization method to minimize the information of a data point to the information that is needed for a given task. To this end, the hierarchical feature extraction model and latent guidance-based data generation model are developed to generate multi-grained data. There are several limitations to the current work.

First, formal guarantees for information minimization concerning a given task need further study. We acknowledge that it is hard to empirically illustrate information minimization since all possible other information would have to be evaluated. Sensing data from Hippo is generated from random noise with the guidance of latent representations of different CAE layers, which contain multi-grained information of raw data. Users can choose the i-th layer to quantify the released sensing information. The method enables different levels of privacy, but these levels are not easily aligned with interpretable privacy requirements. In the future, we will seek provable guarantees and interpretable reconstruction results for information minimization with in-depth development of the learning theory of deep learning. Nevertheless, the success of text-to-image generators such as DALL-E2 [52] proves the ability of data generation models as in Hippo.

Second, there is a risk that reconstructed data could diminish the accuracy of model inference, which is potentially harmful to certain applications (*e.g.*, fall detection, where ideally accuracy should be 100%). Moreover, employing diffusion models to reconstruct an anonymized signal could entirely invalidate any system relying on anomaly detection (*e.g.*, arrhythmia detection). This issue makes the approach unsuitable for such scenarios. We need to continue to explore advanced methods to prevent the degradation of use cases that depend on anomaly detection.

6.3: Outline of Future Work

In the future, I will continue making efforts to enforce security and privacy measures in the interactions between users, devices, environments, and AI. In particular, with the development of Large Models or Foundation Models [255], the generative AI models have shown great ability to simulate the physical world. Meanwhile, these AI models will empower the physical devices with great ability to interact with each other and the surrounding environments. Regarding the security and privacy issues in the interactions of artificial intelligence of things, there are three main directions for further efforts in the future.

Multimodal Federated Learning for Privacy-enhanced Interaction between IoT Devices

The world of the IoT is an important source of multimodal data. IoT devices, ranging from smartphones and wearables to industrial sensors, collect a variety of data types, including images, audio, text, and sensor readings. This diversity is foundational for multimodal learning, providing a rich dataset for comprehensive analysis and insight generation. In addition, IoT devices continuously generate data in real-time, offering a dynamic and evolving dataset for multimodal FL. This continuous data flow is ideal for developing models that adapt and evolve over time, enhancing their accuracy and relevance.

IoT devices often collect sensitive information. Multimodal FL allows for the analysis of this data directly at the source, without needing to centralize it, thereby preserving privacy and reducing data transmission costs. By leveraging data from multiple modalities through FL, IoT systems can make more informed and accurate decisions. For instance, in a smart city context, data from cameras (visual), microphones (audio), and environmental sensors (numerical) can be integrated to optimize traffic management and public safety. Multimodal FL fits naturally with edge computing—a key component of modern IoT frameworks—where data processing occurs closer to where it is generated. This synergy reduces latency, decreases bandwidth usage, and ensures more responsive and autonomous IoT applications.

However, in multimodal FL, data heterogeneity is more challenging because different modalities can have varying data distributions, dimensions, and modalities, making it challenging to integrate and learn from them effectively. Multimodal FL requires sophisticated models that can handle and integrate multiple data types. Designing and training such models in a federated setting adds to the complexity, especially when considering the need to adapt these models to potentially sparse and non-IID (independent and identically distributed) data across nodes. While

FL inherently offers a level of privacy by not sharing raw data, multimodal FL must ensure that combined data modalities do not inadvertently reveal sensitive information. Additionally, robust mechanisms are needed to protect against adversarial attacks that can exploit multimodal data.

Multimodal FL can be applied in various domains, such as healthcare, where different data types can be integrated while respecting patient privacy. We will need to develop new algorithms, including new strategies for data fusion, model training, and communication efficiency, which can contribute to the broader field of machine learning. By leveraging FL, multimodal learning can utilize data from diverse sources without centralizing it, thereby enhancing data privacy and security—a significant advantage in sectors such as healthcare and finance.

Defend against AI-enabled Fraud for Trustworthy Interaction between Users and AI systems

AI-enabled fraud and deception refer to sophisticated methods that use AI technologies to commit or enhance fraudulent activities and deceptive practices. AI-enabled fraud is becoming a serious issue in the interactions between users, devices, and AI because of the ability of AI to simulate the physical world. The adversary can leverage the power of AI to analyze large datasets, identify vulnerabilities, automate deceptive processes, and create realistic forgeries that can deceive individuals, businesses, and even automated systems.

For example, AI can be used to craft more convincing phishing emails and messages by analyzing large datasets to mimic communication styles, creating personalized messages that are more likely to deceive the recipients. Using deep learning, the adversary can synthesize fake images, videos, or voice recordings (deepfakes) that are convincingly real, enabling them to impersonate individuals for unauthorized access or to spread misinformation. In the context of cybersecurity, AI can be used to generate adversarial examples that deceive AI systems, such as manipulating images so that they are misclassified by visual recognition systems.

The sophistication of AI-enabled fraud makes it challenging to detect, as it can often bypass traditional security measures and detection algorithms. In the future, we will develop advanced AI algorithms, leveraging AI itself, to learn and adapt to new fraudulent patterns over time, helping to identify and mitigate AI-enabled fraud. AI can analyze vast amounts of data in real-time, identifying complex patterns and anomalies that might indicate fraudulent activities, thus offering advanced detection capabilities beyond human analysts or traditional methods. By automating the detection process and reducing the incidence of fraud, AI-enabled systems can significantly lower costs related to fraud investigation and losses.

In the future, we will put efforts into enhancing the trustworthiness of interactions between users and AI systems. First, the high-quality and extensive datasets are crucial for training effective AI models. However, accessing diverse and comprehensive fraud-related data while maintaining privacy and security is a significant challenge. AI fraud detection can be integrated with other technologies such as blockchain for enhanced security or IoT for broader data collection, opening up new avenues for innovative solutions. Second, the adversary may continuously evolve the tactics, meaning AI systems need to be adaptable and capable of detecting new and unknown types of fraud, which is a non-trivial task given the adaptive adversaries. Third, balancing the sensitivity of AI models to accurately detect fraud while minimizing false positives (legitimate transactions flagged as fraudulent) and false negatives (fraudulent transactions not detected) is critical to ensure trust and efficiency. Finally, ensuring that fraud detection decisions are explainable and transparent is essential for regulatory compliance and trustworthiness.

AI for Secure Interaction between Users and Devices

The integration of AI for finding and fixing vulnerabilities in critical software within the IoT ecosystem presents a unique set of challenges and opportunities. As IoT devices become more prevalent and integral to our daily lives, ensuring their security is paramount.

The IoT ecosystem is incredibly diverse, with devices varying in computational power, operating systems, and functionalities. Developing AI models that can effectively address the security needs of such a diverse range of devices is a significant challenge. In addition, many IoT devices have limited processing power and memory, which can restrict the deployment of sophisticated AI algorithms directly on the devices. The lack of standardization across IoT devices can complicate the integration of AI solutions for security, as these solutions must be compatible with a wide range of device protocols and architectures. Furthermore, IoT devices often collect sensitive data. Utilizing AI to analyze this data for vulnerabilities must be done in a way that respects user privacy and complies with data protection regulations.

In the future, we will develop lightweight AI models to analyze data from numerous IoT devices in real-time to identify patterns and anomalies that may indicate a security vulnerability or an ongoing attack, allowing for proactive threat mitigation. The AI models can learn the normal behavior of IoT devices and networks, making it possible to detect deviations that may signify a security breach, even in the absence of known malware signatures. The benefits of AI in securing IoT devices apply across various sectors, including healthcare, manufacturing, and smart homes, enhancing overall security and trust in IoT environments.

BIBLIOGRAPHY

[1] Hadi Abdullah, Washington Garcia, Christian Peeters, Patrick Traynor, Kevin RB Butler, and Joseph Wilson. Practical hidden voice attacks against speech and speaker recognition systems. *arXiv preprint arXiv:1904.05734*, 2019.

[2] Mohannad Alhanahnah, Clay Stevens, and Hamid Bagheri. Scalable analysis of interaction threats in iot systems. In *Proceedings of the 29th ACM SIGSOFT international symposium on software testing and analysis*, pages 272–285, 2020.

[3] Mohannad Alhanahnah, Clay Stevens, Bocheng Chen, Qiben Yan, and Hamid Bagheri. Iotcom: Dissecting interaction threats in iot systems. *IEEE Transactions on Software Engineering*, 2022.

[4] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[5] Amazon. Amazon alexa skills. https://www.amazon.com/alexa-skills/b?ie=UTF8&node=13727921011, 2022.

[6] S Abhishek Anand and Nitesh Saxena. Speechless: Analyzing the threat to speech privacy from smartphone motion sensors. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 1000–1017. IEEE, 2018.

[7] Deepali Aneja, Alex Colburn, Gary Faigin, Linda Shapiro, and Barbara Mones. Modeling stylized character expressions via deep learning. In *Asian Conference on Computer Vision*, pages 136–153. Springer, 2016.

[8] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.

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

[10] Home Assistant. Home assistant. https://www.home-assistant.io/, 2022.

[11] Santiago Andrés Azcoitia and Nikolaos Laoutaris. A survey of data marketplaces and their business models. *ACM SIGMOD Record*, 51(3):18–29, 2022.

[12] Zhongjie Ba, Tianhang Zheng, Xinyu Zhang, Zhan Qin, Baochun Li, Xue Liu, and Kui Ren. Learning-based practical smartphone eavesdropping with built-in accelerometer. In *NDSS*, 2020.

[13] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR, 2018.

[14] Billur Barshan and Murat Cihan Yüksek. Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. *The Computer Journal*, 57(11):1649–1667, 2014.

[15] Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel N Kho, and Jennie Rogers. Smcql: Secure query processing for private data networks. *Proc. VLDB Endow.*, 10(6):673–684, 2017.

[16] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[17] Alsallakh Bilal, Amin Jourabloo, Mao Ye, Xiaoming Liu, and Liu Ren. Do convolutional neural networks learn class hierarchy? *IEEE transactions on visualization and computer graphics*, 24(1):152–162, 2017.

[18] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media.", 2009.

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

[20] Luca Bonomi, Liyue Fan, and Hongxia Jin. An information-theoretic approach to individual sequential data sanitization. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 337–346, 2016.

[21] Antoine Boutet, Carole Frindel, Sébastien Gambs, Théo Jourdan, and Rosin Claude Ngueveu. Dysan: Dynamically sanitizing motion sensor data against sensitive inferences through adversarial networks. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 672–686, 2021.

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

[23] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2013.

[24] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[25] Tristan Cazenave. Monte carlo beam search. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):68–72, 2012.

[26] Z Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A Selcuk Uluagac. Sensitive information tracking in commodity iot. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1687–1704, 2018.

[27] Z Berkay Celik, Patrick McDaniel, and Gang Tan. Soteria: Automated iot safety and security analysis. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 147–158, 2018.

[28] Z Berkay Celik, Gang Tan, and Patrick D McDaniel. Iotguard: Dynamic enforcement of security and safety policy in commodity iot. In *NDSS*, 2019.

[29] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[30] Supriyo Chakraborty, Chenguang Shen, Kasturi Rangan Raghavan, Yasser Shoukry, Matt Millar, and Mani Srivastava. ipshield: A framework for enforcing context-aware privacy. In *11th USENIX symposium on networked systems design and implementation (NSDI 14)*, pages 143–156, 2014.

[31] Bocheng Chen, Nikolay Ivanov, Guangjing Wang, and Qiben Yan. Dynamicfl: Balancing communication dynamics and client manipulation for federated learning. In *2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 312–320. IEEE, 2023.

[32] Chen Chen, Ke Sun, and Xinyu Zhang. Exgsense: Toward facial gesture sensing with a sparse near-eye sensor array. In *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021)*, pages 222–237, 2021.

[33] Jiayi Chen, Urs Hengartner, Hassan Khan, and Mohammad Mannan. Chaperone: Real-time locking and loss prevention for smartphones. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 325–342, 2020.

[34] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities. *ACM Computing Surveys (CSUR)*, 54(4):1–40, 2021.

[35] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[36] Yanjiao Chen, Runmin Ou, Zhiyang Li, and Kaishun Wu. Wiface: facial expression recognition using wi-fi signals. *IEEE Transactions on Mobile Computing*, 21(1):378–391, 2020.

[37] Eunyong Cheon, Yonghwan Shin, Jun Ho Huh, Hyoungshick Kim, and Ian Oakley. Gesture authentication for smartphones: Evaluation of gesture password selection policies. In *2020 ieee symposium on security and privacy (sp)*, pages 249–267. IEEE, 2020.

[38] Haotian Chi, Chenglong Fu, Qiang Zeng, and Xiaojiang Du. Delay wreaks havoc on your smart home: Delay-based: Automation interference attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1575–1575. IEEE Computer Society, 2022.

[39] Haotian Chi, Qiang Zeng, Xiaojiang Du, and Jiaping Yu. Cross-app interference threats in smart homes: Categorization, detection and handling. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 411–423. IEEE, 2020.

[40] Seokmin Choi, Yang Gao, Yincheng Jin, Se jun Kim, Jiyang Li, Wenyao Xu, and Zhanpeng Jin. Ppgface: Like what you are watching? earphones can" feel" your facial expressions. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(2):1–32, 2022.

[41] Elizabeth A Clark, J'Nai Kessinger, Susan E Duncan, Martha Ann Bell, Jacob Lahne, Daniel L Gallagher, and Sean F O'Keefe. The facial action coding system for characterization of human affective response to consumer product-based stimuli: a systematic review. *Frontiers in psychology*, 11:920, 2020.

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

[43] Botos Csaba, Xiaojuan Qi, Arslan Chaudhry, Puneet Dokania, and Philip Torr. Multilevel knowledge transfer for cross-domain object detection. *arXiv preprint arXiv:2108.00977*, 2021.

[44] Alexander De Luca, Emanuel Von Zezschwitz, Ngo Dieu Huong Nguyen, Max-Emanuel Maurer, Elisa Rubegni, Marcello Paolo Scipioni, and Marc Langheinrich. Back-of-device authentication on smartphones. In *Proceedings of the sigchi conference on human factors in computing systems*, pages 2389–2398, 2013.

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

[46] Alex Developer. Python functions to count steps in accelerometer data. https://pypi.org/project/pydometer/, 2023.

[47] Apple Developer. Core motion framework reference. https://developer.apple.com/documentation/coremotion, 2023.

[48] Arm Developer. Trust zone. https://www.arm.com/technologies/trustzone-for-cortex-a, 2023.

[49] Google Developer. Activity recognition api. https://developers.google.com/location-context/activity-recognition, 2023.

[50] Google Developer. Android permission. https://developer.android.com/reference/android/Manifest.permission, 2023.

[51] Google Developer. Android sensors. https://developer.android.com/guide/topics/sensors/sensors_overview, 2023.

[52] OpenAI Developer. Ai system that can create realistic images. https://openai.com/dall-e-2/, 2023.

[53] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[54] Wenbo Ding and Hongxin Hu. On the safety of iot device physical interaction control. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 832–846, 2018.

[55] Wenbo Ding, Hongxin Hu, and Long Cheng. Iotsafe: Enforcing safety and security policy with real iot physical interaction discovery. In *Proceedings of the 2021 Network and Distributed Systems Security (NDSS) Symposium*, 2021.

[56] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[57] Alexey Drutsa, Valentina Fedorova, Dmitry Ustalov, Olga Megorskaya, Evfrosiniya Zerminova, and Daria Baidakova. Crowdsourcing practice for efficient data labeling: Aggregation, incremental relabeling, and pricing. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2623–2627, 2020.

[58] Jian Du, Shanghang Zhang, Guanhang Wu, José MF Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

[59] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298, 2017.

[60] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

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

[62] Cynthia Dwork and Guy N Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

[63] P Ekmann. Universal facial expressions in emotion. *Studia Psychologica*, 15(2):140, 1973.

[64] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[65] Hossein Fereidooni, Jan König, Phillip Rieger, Marco Chilese, Bora Gökbakan, Moritz Finke, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. Authentisense: A scalable behavioral biometrics authentication scheme using few-shot learning for mobile platforms. In *Proceedings of the Network and Distributed System Security (NDSS)*, 2023.

[66] Earlence Fernandes, Amir Rahmati, Jaeyeon Jung, and Atul Prakash. Decentralized action integrity for trigger-action iot platforms. In *Proceedings 2018 Network and Distributed System Security Symposium*, 2018.

[67] David J Field. Relations between the statistics of natural images and the response properties of cortical cells. *Josa a*, 4(12):2379–2394, 1987.

[68] Julien Fleureau, Philippe Guillotel, and Izabela Orlac. Affective benchmarking of movies based on the physiological responses of a real audience. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 73–78. IEEE, 2013.

[69] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings 34*, pages 151–164. Springer, 2019.

[70] Chenglong Fu, Qiang Zeng, and Xiaojiang Du. Hawatcher: Semantics-aware anomaly detection for appified smart homes. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[71] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pages 2331–2341, 2020.

[72] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 301–316, 2020.

[73] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[74] Wenbin Gao, Lei Zhang, Wenbo Huang, Fuhong Min, Jun He, and Aiguo Song. Deep neural networks for sensor-based human activity recognition using selective kernel convolution. *IEEE Transactions on Instrumentation and Measurement*, 70:1–13, 2021.

[75] Yang Gao, Yincheng Jin, Seokmin Choi, Jiyang Li, Junjie Pan, Lin Shu, Chi Zhou, and Zhanpeng Jin. Sonicface: Tracking facial expressions using a commodity microphone array. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(4):1–33, 2021.

[76] Zhihui Gao, Ang Li, Dong Li, Jialin Liu, Jie Xiong, Yu Wang, Bing Li, and Yiran Chen. Mom: Microphone based 3d orientation measurement. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 132–144. IEEE, 2022.

[77] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[78] Google. Google assistant. https://assistant.google.com/, 2022.

[79] Michaela Götz, Suman Nath, and Johannes Gehrke. Maskit: Privately releasing user context streams for personalized mobile applications. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 289–300. ACM, 2012.

[80] Xiang Gu, Jian Sun, and Zongben Xu. Spherical space domain adaptation with robust pseudo-label loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9101–9110, 2020.

[81] Yu Gu, Xiang Zhang, Zhi Liu, and Fuji Ren. Wife: Wifi and vision based intelligent facial-gesture emotion recognition. *arXiv preprint arXiv:2004.09889*, 2020.

[82] Hanqing Guo, Chenning Li, Lingkun Li, Zhichao Cao, Qiben Yan, and Li Xiao. Nec: Speaker selective cancellation via neural enhanced ultrasound shadowing. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 355–366. IEEE, 2022.

[83] Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu, and Shi-Min Hu. Beyond self-attention: External attention using two linear layers for visual tasks. *arXiv preprint arXiv:2105.02358*, 2021.

[84] Omid Hajihassnai, Omid Ardakanian, and Hamzeh Khazaei. Obscurenet: Learning attribute-invariant latent representation for anonymizing sensor data. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, pages 40–52, 2021.

[85] Feng Han, Lan Zhang, Hanwen Feng, Weiran Liu, and Xiangyang Li. Scape: Scalable collaborative analytics system on private database with malicious security. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1740–1753. IEEE, 2022.

[86] Feng Han, Lan Zhang, Xuanke You, Guangjing Wang, and Xiang-Yang Li. Shad: Privacy-friendly shared activity detection and data sharing. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 109–117. IEEE, 2019.

[87] Jianping He, Lin Cai, and Xinping Guan. Preserving data-privacy with added noises: Optimal estimation and privacy analysis. *IEEE Transactions on Information Theory*, 64(8):5677–5690, 2018.

[88] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[89] Xi He, Graham Cormode, Ashwin Machanavajjhala, Cecilia Procopiuc, and Divesh Srivastava. Dpt: differentially private trajectory synthesis using hierarchical reference systems. *Proceedings of the VLDB Endowment*, 8(11):1154–1165, 2015.

[90] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[91] Eran Hof, Amichai Sanderovich, Mohammad Salama, and Evyatar Hemo. Face verification using mmwave radar sensor. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 320–324. IEEE, 2020.

[92] Pengfei Hu, Hui Zhuang, Panneer Selvam Santhalingam, Riccardo Spolaor, Parth Pathak, Guoming Zhang, and Xiuzhen Cheng. Accear: Accelerometer acoustic eavesdropping with unconstrained vocabulary. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1530–1530. IEEE Computer Society, 2022.

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

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

[95] Shao-Lun Huang, Xiangxiang Xu, Lizhong Zheng, and Gregory W Wornell. An information theoretic interpretation to deep neural networks. In *2019 IEEE international symposium on information theory (ISIT)*, pages 1984–1988. IEEE, 2019.

[96] Xiao Huang, Qingquan Song, Fan Yang, and Xia Hu. Large-scale heterogeneous feature embedding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3878–3885, 2019.

[97] IFTTT. Ifttt applets. https://ifttt.com/explore/, 2022.

[98] IFTTT. Ifttt applets. https://ifttt.com/explore/, 2022.

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

[100] Yusuke Iwasawa, Kotaro Nakayama, Ikuko Yairi, and Yutaka Matsuo. Privacy issues regarding the application of dnns to activity-recognition using wearables and its countermeasures by use of adversarial training. In *IJCAI*, pages 1930–1936, 2017.

[101] Ankita Jain and Vivek Kanhangad. Investigating gender recognition in smartphones using accelerometer and gyroscope sensor readings. In *2016 international conference on computational techniques in information and communication technologies (ICCTICT)*, pages 597–602. IEEE, 2016.

[102] Wenchao Jiang and Zhaozheng Yin. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1307–1310, 2015.

[103] Yupeng Jiang, Yong Li, Yipeng Zhou, and Xi Zheng. Sybil attacks and defense on differential privacy based federated learning. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 355–362. IEEE, 2021.

[104] Kleomenis Katevas. Sensingkit. https://www.sensingkit.org/, 2023.

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

[106] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[107] Jinkyu Kim, Heonseok Ha, Byung-Gon Chun, Sungroh Yoon, and Sang K Cha. Collaborative analytics for data silos. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 743–754. IEEE, 2016.

[108] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[109] Palanivel A Kodeswaran, Ravi Kokku, Sayandeep Sen, and Mudhakar Srivatsa. Idea: A system for efficient failure management in smart iot environments. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 43–56, 2016.

[110] Sarah Martina Kolly, Roger Wattenhofer, and Samuel Welten. A personal touch: Recognizing users based on touch screen behavior. In *Proceedings of the third international workshop on sensing applications on mobile phones*, pages 1–5, 2012.

[111] Harold William Kuhn and Albert William Tucker. *Contributions to the Theory of Games*, volume 2. Princeton University Press, 1953.

[112] Abhishek Kumar, Prasanna Sattigeri, and Tom Fletcher. Semi-supervised learning with gans: Manifold invariance with improved inference. *Advances in neural information processing systems*, 30, 2017.

[113] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209, 2012.

[114] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of experimental social psychology*, 49(4):764–766, 2013.

[115] Ang Li, Yixiao Duan, Huanrui Yang, Yiran Chen, and Jianlei Yang. Tiprdc: task-independent privacy-respecting data crowdsourcing framework for deep learning with anonymized intermediate representations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 824–832, 2020.

[116] Anran Li, Lan Zhang, Junhao Wang, Feng Han, and Xiangyang Li. Privacy-preserving efficient federated-learning model debugging. *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[117] Anran Li, Lan Zhang, Junhao Wang, Juntao Tan, Feng Han, Yaxuan Qin, Nikolaos M Freris, and Xiang-Yang Li. Efficient federated-learning model debugging. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 372–383. IEEE, 2021.

[118] Chenning Li, Zhichao Cao, and Yunhao Liu. Deep ai enabled ubiquitous wireless sensing: A survey. *ACM Computing Surveys (CSUR)*, 54(2):1–35, 2021.

[119] Chenning Li, Manni Liu, and Zhichao Cao. Wihf: enable user identified gesture recognition with wifi. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 586–595. IEEE, 2020.

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

[121] Dong Li, Jialin Liu, Sunghoon Ivan Lee, and Jie Xiong. Lasense: Pushing the limits of fine-grained activity sensing using acoustic signals. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(1):1–27, 2022.

[122] Kaiyu Li, Guoliang Li, Yong Wang, Yan Huang, Zitao Liu, and Zhongqin Wu. Crowdrl: an end-to-end reinforcement learning framework for data labelling. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 289–300. IEEE, 2021.

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

[124] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 106–115. IEEE, 2007.

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

[126] Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *IEEE transactions on affective computing*, 2020.

[127] Tao Li, Yexi Jiang, Chunqiu Zeng, Bin Xia, Zheng Liu, Wubai Zhou, Xiaolong Zhu, Wentao Wang, Liang Zhang, Jun Wu, et al. Flap: An end-to-end event log analysis platform for system management. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1547–1556, 2017.

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

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

[130] Cihang Liu, Lan Zhang, Zongqian Liu, Kebin Liu, Xiangyang Li, and Yunhao Liu. Lasagna: towards deep hierarchical understanding and searching over mobile sensing data. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 334–347. ACM, 2016.

[131] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[132] Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Haiyang Yu, Zhao Xu, Jingtun Zhang, Yi Liu, Keqiang Yan, Haoran Liu, Cong Fu, Bora M Oztekin, Xuan Zhang, and Shuiwang Ji. DIG: A turnkey library for diving into graph deep learning research. *Journal of Machine Learning Research*, 22(240):1–9, 2021.

[133] Sicong Liu, Junzhao Du, Anshumali Shrivastava, and Lin Zhong. Privacy adversarial network: representation learning for mobile data privacy. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(4):1–18, 2019.

[134] Wenyan Liu, Junhong Cheng, Xiaoling Wang, Xingjian Lu, and Jianwei Yin. Hybrid differential privacy based federated learning for internet of things. *Journal of Systems Architecture*, 124:102418, 2022.

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

[136] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

[137] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.

[138] Mitja Luštrek and Boštjan Kaluža. Fall detection and activity recognition with machine learning. *Informatica*, 33(2), 2009.

[139] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[140] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. Protecting sensory data against sensitive inferences. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*, pages 1–6, 2018.

[141] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. Mobile sensor data anonymization. In *Proceedings of the international conference on internet of things design and implementation*, pages 49–58, 2019.

[142] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. Mobile sensor data anonymization. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, IoTDI '19, pages 49–58, New York, NY, USA, 2019. ACM.

[143] Mohammad Malekzadeh, Richard G Clegg, and Hamed Haddadi. Replacement autoencoder: A privacy-preserving algorithm for sensory data analysis. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 165–176. IEEE, 2018.

[144] Wenguang Mao, Mei Wang, and Lili Qiu. Aim: Acoustic imaging on a mobile. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 468–481, 2018.

[145] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

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

[147] Maryam Mehrnezhad, Ehsan Toreini, Siamak F Shahandashti, and Feng Hao. Touchsignatures: identification of user touch actions and pins based on mobile sensor data via javascript. *Journal of Information Security and Applications*, 26:23–38, 2016.

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

[149] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing speech from gyroscope signals. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 1053–1067, 2014.

[150] Kato Mivule. Utilizing noise addition for data privacy, an overview. *arXiv preprint arXiv:1309.3958*, 2013.

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

[152] Reiichiro Nakano. Scikit-plot. https://github.com/reiinakano/scikit-plot, 2017.

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

[154] Dang Tu Nguyen, Chengyu Song, Zhiyun Qian, Srikanth V Krishnamurthy, Edward JM Colbert, and Patrick McDaniel. Iotsan: Fortifying the safety of iot systems. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*, pages 191–203, 2018.

[155] Jingping Nie, Yigong Hu, Yuanyuting Wang, Stephen Xia, and Xiaofan Jiang. Spiders: Low-cost wireless glasses for continuous in-situ bio-signal acquisition and emotion recognition. In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 27–39. IEEE, 2020.

[156] Tin Lay Nwe, Foo Say Wei, and Liyanage C De Silva. Speech based emotion classification. In *Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology. TENCON 2001 (Cat. No. 01CH37239)*, volume 1, pages 297–301. IEEE, 2001.

[157] OpenAI et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[158] Atila Orhon, Michael Siracusa, and Aseem Wadhwa. Stable diffusion with core ml on apple silicon, 2022.

[159] Charlie Osborne. Surveillance cameras sold on amazon infected with malware. https://www.zdnet.com/article/amazon-surveillance-cameras-infected-with-malware/, 2016.

[160] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Kleomenis Katevas, Hamed Haddadi, and Hamid R Rabiee. Deep private-feature extraction. *IEEE Transactions on Knowledge and Data Engineering*, 32(1):54–66, 2018.

[161] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. Clusterfl: a similarity-aware federated learning system for human activity recognition. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 54–66, 2021.

[162] Adam Paszke and Sam Gross. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[163] Jayashree V Patil and Preeti Bailke. Real time facial expression recognition using realsense camera and ann. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 2, pages 1–6. IEEE, 2016.

[164] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[165] Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. Differentially private federated knowledge graphs embedding. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1416–1425, 2021.

[166] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[167] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[168] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10772–10781, 2019.

[169] Thomas Prätzlich, Jonathan Driedger, and Meinard Müller. Memory-restricted multiscale dynamic time warping. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 569–573. IEEE, 2016.

[170] Laavanya Rachakonda, Saraju P Mohanty, and Elias Kougianos. cstick: a calm stick for fall prediction, detection and control in the iomt framework. In *Internet of Things. Technology and Applications: 4th IFIP International Cross-Domain Conference, IFIPIoT 2021, Virtual Event, November 4–5, 2021, Revised Selected Papers*, pages 129–145. Springer, 2022.

[171] Kasturi Rangan Raghavan, Supriyo Chakraborty, Mani Srivastava, and Harris Teague. Override: A mobile privacy framework for context-driven perturbation and synthesis of sensor data streams. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones*, pages 1–5, 2012.

[172] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[173] Zhongzheng Ren, Yong Jae Lee, and Michael S Ryoo. Learning to anonymize faces for privacy preserving action detection. In *Proceedings of the european conference on computer vision (ECCV)*, pages 620–636, 2018.

[174] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999.

[175] Delphine Rommel, JL Nandrino, M Jeanne, R Logier, et al. Heart rate variability analysis as an index of emotion regulation processes: interest of the analgesia nociception index (ani). In *2012 Annual international conference of the IEEE engineering in medicine and biology society*, pages 3432–3435. IEEE, 2012.

[176] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[177] Nirupam Roy, Haitham Hassanieh, and Romit Roy Choudhury. Backdoor: Making microphones hear inaudible sounds. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 2–14, 2017.

[178] Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

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

[180] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[181] Scrapy. Web crawling framework. https://scrapy.org/, 2022.

[182] Gökhan Şengül, Murat Karakaya, Sanjay Misra, Olusola O Abayomi-Alli, and Robertas Damaševičius. Deep learning based fall detection using smartwatches for healthcare applications. *Biomedical Signal Processing and Control*, 71:103242, 2022.

[183] Tak-Wai Shen, Hong Fu, Junkai Chen, WK Yu, CY Lau, WL Lo, and Zheru Chi. Facial expression recognition using depth map estimation of light field camera. In *2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–4. IEEE, 2016.

[184] Michael Sherman, Gradeigh Clark, Yulong Yang, Shridatt Sugrim, Arttu Modig, Janne Lindqvist, Antti Oulasvirta, and Teemu Roos. User-generated free-form gestures for authentication: Security and memorability. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 176–189, 2014.

[185] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[186] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[187] Diksha Shukla and Vir V Phoha. Stealing passwords by observing hands movement. *IEEE Transactions on Information Forensics and Security*, 14(12):3086–3101, 2019.

[188] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[189] SmartThings. Smartthings developer. https://smartthings.developer.samsung.com/docs/api-ref/capabilities.html, 2022.

[190] SmartThings. Smartthings developer. https://smartthings.developer.samsung.com/docs/api-ref/capabilities.html, 2022.

[191] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[192] Congzheng Song and Vitaly Shmatikov. Overlearning reveals sensitive attributes. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[193] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[194] Xingzhe Song, Kai Huang, and Wei Gao. Facelistener: Recognizing human facial expressions via acoustic sensing on commodity headphones. In *21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2022.

[195] Spacy. Industrial-strength natural language processing. https://spacy.io/, 2022.

[196] Fan-Yun Sun, Jordon Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.

[197] Milijana Surbatovich, Jassim Aljuraidan, Lujo Bauer, Anupam Das, and Limin Jia. Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of ifttt recipes. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1501–1510, 2017.

[198] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

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

[200] Yuan Tian, Nan Zhang, Yueh-Hsun Lin, XiaoFeng Wang, Blase Ur, Xianzheng Guo, and Patrick Tague. Smartauth: User-centered authorization for the internet of things. In *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[201] Marcos Tileria, Jorge Blasco, and Guillermo Suarez-Tangil. {WearFlow}: Expanding information flow analysis to companion apps in wear {OS}. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 63–75, 2020.

[202] Yongxin Tong, Xuchen Pan, Yuxiang Zeng, Yexuan Shi, Chunbo Xue, Zimu Zhou, Xiaofei Zhang, Lei Chen, Yi Xu, Ke Xu, et al. Hu-fu: efficient and secure spatial queries over data federation. *Proceedings of the VLDB Endowment*, 15(6):1159, 2022.

[203] Rahmadi Trimananda, Seyed Amir Hossein Aqajari, Jason Chuang, Brian Demsky, Guoqing Harry Xu, and Shan Lu. Understanding and automatically detecting conflicting interactions between smart home iot applications. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1215–1227, 2020.

[204] Yu-Chih Tung, Duc Bui, and Kang G Shin. Cross-platform support for rapid development of mobile acoustic sensing applications. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 455–467, 2018.

[205] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L Littman. 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*, pages 3227–3231, 2016.

[206] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

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

[208] Dhruv Verma, Sejal Bhalla, Dhruv Sahnan, Jainendra Shukla, and Aman Parnami. Expressear: Sensing fine-grained facial expressions with earables. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(3):1–28, 2021.

[209] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[210] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, et al. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 1, 2023.

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

[212] Chen Wang, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. Friend or foe? your wearable devices reveal your personal pin. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 189–200, 2016.

[213] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2017.

[214] Guangjing Wang, Hanqing Guo, Anran Li, Xiaorui Liu, and Qiben Yan. Federated iot interaction vulnerability analysis. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023.

[215] Guangjing Wang, Nikolay Ivanov, Bocheng Chen, Qi Wang, ThanhVu Nguyen, and Qiben Yan. Graph learning for interactive threat detection in heterogeneous smart home rule data. *Proceedings of the ACM on Management of Data*, 1(1):1–27, 2023.

[216] Guangjing Wang, Qiben Yan, Shane Patrarungrong, Juexing Wang, and Huacheng Zeng. Facer: Contrastive attention based expression recognition via smartphone earpiece speaker. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2023.

[217] Guangjing Wang, Lan Zhang, Zhi Yang, and Xiang-Yang Li. Socialite: Social activity mining and friend auto-labeling. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2018.

[218] Haiming Wang, Zhikun Zhang, Tianhao Wang, Shibo He, Michael Backes, Jiming Chen, and Yang Zhang. PrivTrace: Differentially Private Trajectory Synthesis by Adaptive Markov Model. In *USENIX Security*, 2023.

[219] Hao Wang and Zhengquan Xu. Cts-dp: Publishing correlated time-series data via differential privacy. *Knowledge-Based Systems*, 122:167–179, 2017.

[220] Junhao Wang, Lan Zhang, Anran Li, Xuanke You, and Haoran Cheng. Efficient participant contribution evaluation for horizontal and vertical federated learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 911–923. IEEE, 2022.

[221] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

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

[223] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. Fear and logging in the internet of things. In *Network and Distributed Systems Symposium*, 2018.

[224] Yuanda Wang, Hanqing Guo, and Qiben Yan. Ghosttalk: Interactive attack on smartphone voice system through power line. *arXiv preprint arXiv:2202.02585*, 2022.

[225] Yue Wang, Xintao Wu, and Donghui Hu. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops*, volume 1558, pages 0090–6778, 2016.

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

[227] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[228] Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021.

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

[230] Yadong Xie, Fan Li, Yue Wu, Huijie Chen, Zhiyuan Zhao, and Yu Wang. Teethpass: Dental occlusion-based user authentication via in-ear acoustic sensing. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 1789–1798. IEEE, 2022.

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

[232] Xiangxiang Xu, Shao-Lun Huang, Lizhong Zheng, and Gregory W Wornell. An information theoretic interpretation to deep neural networks. *Entropy*, 24(1):135, 2022.

[233] Xiangyu Xu, Jiadi Yu, Yingying Chen, Yanmin Zhu, Linghe Kong, and Minglu Li. Breathlistener: Fine-grained breathing monitoring in driving environments utilizing acoustic signals. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 54–66, 2019.

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

[235] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Ali Ahmadzadeh, Xinyu Xing, and Gang Wang. {CADE}: Detecting and explaining concept drift samples for security applications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2327–2344, 2021.

[236] Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, and Wenbin Zhang. Semi-supervised log-based anomaly detection via probabilistic label estimation. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1448–1460. IEEE, 2021.

[237] Doguhan Yeke, Muhammad Ibrahim, Güliz Seray Tuncay, Habiba Farrukh, Abdullah Imran, Antonio Bianchi, and Z Berkay Celik. Wear's my data? understanding the cross-device runtime permission model in wearables. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 77–77. IEEE Computer Society, 2023.

[238] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32:9240, 2019.

[239] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

[240] Hao Yu, Sen Yang, and Shenghuo Zhu. 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*, volume 33, pages 5693–5700, 2019.

[241] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*, pages 12241–12252. PMLR, 2021.

[242] Mu Yuan, Lan Zhang, Xiang-Yang Li, and Hui Xiong. Comprehensive and efficient data labeling via adaptive model scheduling. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1858–1861. IEEE, 2020.

[243] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *6th international conference on mobile computing, applications and services*, pages 197–205. IEEE, 2014.

[244] Anna Zhadan. Report: your smart home devices. https://cybernews.com/security/report-your-smart-home-devices-may-suffer-from-10000-hacking-attempts-every-week/, 2021.

[245] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 103–117, 2017.

[246] Xinyang Zhang, Shouling Ji, and Ting Wang. Differentially private releasing via deep generative model (technical report). *arXiv preprint arXiv:1801.01594*, 2018.

[247] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy. In *ACM CCS*, 2018.

[248] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. PrivSyn: Differentially Private Data Synthesis. In *USENIX Security*, 2021.

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

[250] Mingmin Zhao, Fadel Adib, and Dina Katabi. Emotion recognition using wireless signals. In *Proceedings of the 22nd annual international conference on mobile computing and networking*, pages 95–108, 2016.

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

[252] Tianhang Zheng and Baochun Li. Infocensor: An information-theoretic framework against sensitive attribute inference and demographic disparity. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 437–451, 2022.

[253] Yue Zheng, Yi Zhang, Kun Qian, Guidong Zhang, Yunhao Liu, Chenshu Wu, and Zheng Yang. Zero-effort cross-domain gesture recognition with wi-fi. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 313–325, 2019.

[254] Bing Zhou, Jay Lohokare, Ruipeng Gao, and Fan Ye. Echoprint: Two-factor authentication using acoustics and vision on smartphones. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 321–336, 2018.

[255] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023.

[256] Chuanxin Zhou, Yi Sun, Degang Wang, and Qi Gao. Fed-fi: Federated learning malicious model detection method based on feature importance. *Security and Communication Networks*, 2022, 2022.

[257] Pengpeng Zhou, Yang Wang, Zhenyu Li, Xin Wang, Gareth Tyson, and Gaogang Xie. Logsayer: Log pattern-driven cloud component anomaly diagnosis with machine learning. In *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2020.

[258] Zhijun Zhou, Yingtian Zhang, Xiaojing Yu, Panlong Yang, Xiang-Yang Li, Jing Zhao, and Hao Zhou. Xhar: Deep domain adaptation for human activity recognition with smart devices. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2020.

[259] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.