

# **Towards Private and Accurate IoT Applications**

by

**Jiechao Gao**

A dissertation document submitted in partial fulfillment  
of the requirements for the degree of  
Doctor of Philosophy  
in

Department of Computer Science  
University of Virginia  
July, 2024

Doctoral Committee:

Prof. Bradford Campbell  
Prof. Yangfeng Ji, Committee Chair  
Prof. Lu Feng  
Prof. Cong Shen  
Prof. Jorge Ortiz

© Copyright by Jiechao Gao 2024  
All Rights Reserved

## APPROVAL SHEET

This  
Dissertation  
is submitted in partial fulfillment of the requirements  
for the degree of  
Doctor of Philosophy

Author: Jiechao Gao

This Dissertation has been read and approved by the examining committee:

Advisor: **Dr. Bradford Campbell**

Advisor:

Committee Member: **Dr. Yangfeng Ji**

Committee Member: **Dr. Lu Feng**


Committee Member: **Dr. Cong Shen**

Committee Member: **Dr. Jorge Ortiz**

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:



Jennifer L. West, School of Engineering and Applied Science

August 2024

## ABSTRACT

Over the past years, the fast-growing trend of Internet of Things (IoT) is bringing millions of new smart devices and sensors into homes, office buildings and industries. These smart devices and sensors enable smart IoT applications (e.g., energy prediction, activity recognition, etc.) to increase the quality and efficiency of our lives. To achieve promising performance for smart IoT applications, it requires massive data from different users and sensors to guarantee the performance due to machine learning and deep learning purposes. However, the edge devices of IoT applications often collect and store only limited data, which is insufficient for training modern learning models. Collaboratively training sets steps to achieve better application performance among different devices, while introducing the concern of data privacy. On the other hand, directly applying privacy-preserving techniques such as differential privacy can dramatically degrade the performance of IoT applications.

In this dissertation, we aim to achieve privacy-first smart IoT applications while ensuring their accurate performance for multi-user and multi-sensor scenarios. First, we propose Personalized Federated Deep Reinforcement Learning (PFDRL), a system that helps local users to achieve private and accurate energy management. PFDRL replaces the central server with decentralized federated learning (DFL) framework and enables a personalized federated reinforcement learning to tackle the standby energy reduction in residential building. Next, we present `Atlas`, a private and accurate personalized federated local differential privacy (LDP) framework for IoT applications. We first design a layer-sharing mechanism called layer importance mask to separate the local model into global and personalized layers. Second, we design a weighted LDP mechanism and add noise to the global layers before transmitting them to the federated learning framework for aggregation. Third, we combine local personalized layers and aggregated global layers to perform IoT tasks. Finally, we introduce `PrivateHub`, a system that utilizes contrastive learning with diffusion models for synthetic data generation in multi-sensor scenarios. `PrivateHub` helps to prevent the private applications' identification from multi-sensor environments while ensuring the accurate performance of the non-private applications.

## ACKNOWLEDGEMENTS

I want to extend my deepest gratitude to all those who supported and guided me throughout the completion of my Ph.D. journey over the past six years. The contributions of many individuals have been crucial in shaping my research and personal development. I particularly want to show my deepest appreciation to my advisor and committee members, whose unwavering support and expertise made this achievement possible.

I would like to extend my deepest gratitude to my Ph.D. advisor, Dr. Brad Campbell. His unwavering support, expert guidance, and relentless dedication have been fundamental to my academic journey. Throughout the years, Dr. Campbell has not only provided invaluable insights that refined my research but also encouraged me to think critically and push the boundaries of my knowledge. His mentorship has been a cornerstone of my intellectual development, shaping my abilities and inspiring me to strive for excellence. For his commitment, patience, and encouragement, I am profoundly grateful and will always cherish the impact he has had on both my academic and personal growth. I also sincerely thank the members of my dissertation committee: Dr. Yangfeng Ji, Dr. Lu Feng, Dr. Cong Shen, and Dr. Jorge Ortiz. Their expertise, constructive critiques, and valuable suggestions significantly enriched my research and broadened my understanding of the field. I deeply appreciate their commitment to reviewing my work and providing invaluable insights that greatly enhanced the quality of my dissertation.

I also deeply thank my fellow colleagues and friends who provided essential support, engaging discussions, and a stimulating academic environment. Their camaraderie made the challenging journey of pursuing a Ph.D. more enjoyable and fulfilling. I owe a debt of gratitude to the faculty and staff of the College of Engineering at the University of Virginia for their continuous efforts to maintain a supportive research environment, which greatly contributed to my academic success.

I express my gratitude to my family for their love, understanding, and encouragement throughout my academic journey. Their belief in my abilities provided a constant source of inspiration and motivation. Each of these individuals played a crucial role in my academic and personal growth,

and for their contributions, I remain eternally grateful.

*To my parents and my wife, for their continuous support over the years.*

# Table of Contents

<b>List of Figures</b> . . . . .	11
<b>List of Tables</b> . . . . .	14
<b>Chapter 1: Introduction</b> . . . . .	15
1.1 Thesis Statement . . . . .	16
1.2 Contributions of this Dissertation . . . . .	16
<b>Chapter 2: PFDRL: Personalized Federated Deep Reinforcement Learning for Residential Energy Management</b> . . . . .	19
2.1 Related Work . . . . .	22
2.1.1 Load Forecasting & Energy Management . . . . .	22
2.1.2 Energy Data Privacy & Federated Learning . . . . .	22
2.1.3 Energy Management Personalization . . . . .	23
2.2 System Design . . . . .	23
2.2.1 System Overview . . . . .	23
2.2.2 Decentralized Federated Learning for Load Forecasting . . . . .	25
2.2.3 Personalized Federated DRL For Energy Management . . . . .	27
2.2.3.1 DRL for energy management: . . . . .	28
2.2.3.2 Federated Personalization for Reinforcement Learning: . . . . .	31
2.2.4 Datasets and Experiment Settings . . . . .	32
2.2.5 Compared Methods . . . . .	34
2.2.6 Performance Metrics . . . . .	35
2.3 Experimental Results . . . . .	35
2.4 Case study . . . . .	39



2.5	Conclusion	40
<b>Chapter 3: Atlas: Ensuring Accuracy for Privacy-Preserving IoT Applications</b>		
3.1	Background and motivation	45
3.1.1	Privacy Preserving for FL	46
3.1.2	The Need for Personalization in FL	46
3.2	Related Work	47
3.3	System Design	48
3.3.1	System Overview	48
3.3.2	Model Layer Selection and Layer Importance Mask	49
3.3.3	Dynamic Local Differential Privacy	51
3.3.4	Central Server Aggregation	55
3.3.5	Personalized Model Generation	56
3.4	Evaluation	57
3.4.1	Applications and Datasets	57
3.4.2	Experimental Setup and Evaluation Metrics	60
3.4.3	Results	61
3.4.3.1	RQ1: Performance Analysis on real-world IoT data	61
3.4.3.2	RQ2: Performance analysis on scalability.	62
3.4.3.3	RQ3: Impact of Different LDP Parameter $\epsilon$	63
3.4.3.4	RQ4: Impact on New Users	63
3.4.3.5	RQ5: Communication Efficiency	64
3.5	Conclusion	64
<b>Chapter 4: PrivateHub: Contrastive Diffusion Model for Private Multi-Sensor Scenario Data Generation</b>		
4.1	Related Work	73
4.1.1	Data Privacy	73
4.1.2	Generative Models	74
4.2	Motivation	75
4.2.1	The New Privacy Challenge	75
4.2.2	Beneficial for IoT Data training among users	76
4.3	System Design	79

	10
4.3.1	Design Fashion of Pre-training and Fine-tuning . . . . . 80
4.3.2	App-Conditioned Pre-training . . . . . 80
4.3.2.1	Optimization Objective . . . . . 82
4.3.3	App-Aware Fine-tuning . . . . . 84
4.3.3.1	App-Aware Feature Discrimination . . . . . 85
4.4	Evaluation . . . . . 87
4.4.1	Applications and Datasets . . . . . 87
4.4.2	Experimental Setup and Evaluation Metrics . . . . . 88
4.4.3	Results . . . . . 89
4.4.3.1	RQ1: Performance of synthetic data generation . . . . . 89
4.4.3.2	RQ2: Performance of synthetic data generation with switched private and non-private activities . . . . . 91
4.4.3.3	RQ3: Performance of synthetic data generation with original purpose 92
4.4.3.4	RQ4: Performance of PrivateHub on missing application labels 94
4.5	Conclusion . . . . . 94
<b>Chapter 5: Conclusion</b>	. . . . . 96
5.0.1	Limitations and Future Work . . . . . 97
<b>References</b>	. . . . . 111

# List of Figures

2.1	Results on daily energy consumption of 5 devices in a day. We calculate the standby energy usage and active usage following the setup: Amazon TV Stick, Sony Smart TV, Smart Bulb, and Bose sound system are active for 8 hours and standby for 16 hours, Xbox is active for 3 hours and standby for 21 hours [8], [9]. . . . .	20
2.2	Overview of the PFDRL Framework: ① Local energy data collection and training for standby energy identification; ② Parameters broadcasting at certain time frequency $\beta$ between each residence; ③ Local testing with updated model; ④ Feed load forecasting result together with real-time energy value as deep reinforcement learning environment; ⑤ State observation; ⑥ Reward calculation; ⑦ Action selection to determine device mode; ⑧ Divide neural networks inside the deep reinforcement learning models as base and personalization layers using optimization parameter $\alpha$ . Broadcast base layers at a certain time frequency $\gamma$ and keep the personalization layers locally. Each residence will use the updated PFDRL framework to perform energy management locally. . . . .	24
2.3	Comparison methods. . . . .	34
2.4	System Parameters . . . . .	36
2.5	Prediction Accuracy Comparison . . . . .	37
2.6	Prediction Accuracy Comparison with Different Experiment Settings . . . . .	37
2.7	Saved energy per residence via days. . . . .	38
2.8	Saved monetary cost per residence via months. . . . .	38

	12
2.9 Saved energy per residence in a day. . . . .	39
2.10 System performance in personalization. . . . .	39
2.11 Case Study . . . . .	40
3.1 High-level Atlas Architecture . . . . .	42
3.2 Comparison of Accuracy between Noise-free and GLDP FedAVG in Different Model Configurations. . . . .	47
3.3 The Overview of Atlas Framework: ① Local IoT data training with layer importance mask for local and global layers selection; ② Dynamically adding LDP to each selected global layers by the importance of each layer. ③ Transmitting selected global layers with LDP from the devices to the central server. ④ Central server layer by layer aggregation. ⑤ Sending back updated layer information to local clients. ⑥ Personalized model generation. . . . .	49
3.4 Personalization Layer Selection via Layer Importance Mask . . . . .	50
3.5 Dynamically adjust model weights to LDP mechanism. . . . .	53
3.6 Dynamic LDP with Parameter Shuffling . . . . .	54
3.7 Atlas Model Aggregation . . . . .	55
3.8 Atlas Personalized Model Generation . . . . .	57
3.9 Performance comparison on accuracy and $R^2$ between traditional methods, non-privacy-preserving FL methods, LDP-based FL methods and Atlas . . . . .	65
3.10 Model performance comparison on variate of scalability between non-privacy-preserving FL methods, LDP-based FL methods and Atlas. . . . .	66
3.11 Model performance comparison on variate of privacy budget $\epsilon$ between LDP-based FL methods and Atlas. . . . .	67
3.12 Model performance comparison on adding new users in Atlas. . . . .	68

3.13 Model performance comparison on communication efficiency between FL methods and Atlas. . . . .	69
4.1 High-level architecture of PrivateHub . . . . .	71
4.2 Example of sensor data collection. . . . .	77
4.3 Confusion matrix of our preliminary study . . . . .	77
4.4 Training setups of PrivateHub . . . . .	78
4.5 The overall pipeline of the App-Conditioned Pre-training (ACP) stage. . . . .	81
4.6 The overall pipeline of the App-Aware Fine-tuning (AAF) stage. . . . .	83
4.7 The overall pipeline of the App-Aware Feature Discrimination (AAFD). Herein, the model in gray represents the off-the-shelf activity classifier; “FC” denotes the fully-connected layer, which is normally a single linear projection layer to project the final latent representations into a probability distribution. . . . .	86
4.8 Performance comparison on switch private and non-private activities (Y-axis is accuracy) . . . . .	93
4.9 Performance comparison on CO <sub>2</sub> and temperature prediction (Y-axis is accuracy) . . . . .	94
4.10 Performance comparison on missing application labels (Y-axis is accuracy) . . . . .	95

# List of Tables

2.1	Reward Function . . . . .	30
3.1	Comparing <code>Atlas</code> with existing FL approaches in IoT scenario. . . . .	45
3.2	Dataset Statistics. . . . .	59
4.1	Characteristics of sensors . . . . .	77
4.2	Performance Metrics Comparison . . . . .	90

# Chapter 1

## INTRODUCTION

The trend of Internet of Things (IoT) is bringing in millions of new smart devices such as smart-phones, sensors and wearables in our physical environment to increase the quality of human life. The continuously growing number of smart devices and users provide massive amount of IoT sensing data along with the recent advancements in deep learning techniques have shown great potential for smart Internet of Things (IoT) applications. To achieve these exciting IoT applications, the users often need to (1) aggregate data from more users to build a robust model, or (2) utilize different types of sensors. However, a growing concern with these data aggregation is data privacy. For aggregating data from different users: Since the edge devices of IoT applications often collect and store only limited data, which is insufficient for training modern deep learning models. Collaborative training methods such as cloud computing and federated learning set steps to build robust models for IoT applications by aggregating data from more users. However, these methods bring the concern of data privacy (e.g., untrusted central server, model inversion). According to the McAfee survey [1], 52% of users have experienced stolen data while using cloud service. As a result, the number of people willing to share their personal data fell from 41% to 31% from 2018 to 2019 [2]. On the other hand, directly applying privacy-preserving techniques such as differential privacy and local differential privacy can dramatically degrade the accurate performance of IoT applications. For utilizing different types of sensors: Multiple types of sensor fusion such as lights, doors, speakers, appliances, CO<sub>2</sub>, etc. are installed and controlled by the IoT systems in our homes and work spaces, all of which enhance our quality of life [3]. However, such a sensor fusion scenario can reflect private activities. In our previous work [4], we discovered that by combining IoT devices with CO<sub>2</sub> sensors and humidity sensor, we can successfully identify whether the room is occupied or not. The concern in this multi-sensor setting is that, we aim to achieve accurate performance for the original purpose of these sensors and prevent to detect these latent and private activities. Such a trade-off

between data privacy and IoT application accuracy has been a bottleneck for the development of modern IoT applications.

To address these aforementioned concerns, in this dissertation, we explore three possible directions to achieve privacy-first IoT applications while ensuring a comparable accuracy performance to non-privacy-preserving methods. Our first approach targets the group of local users who shares similar user patterns and wish to train their data in a controlled and secured environment without malicious or honest but curious cloud service. Our second approach applies to global users with massive kinds of IoT applications that use malicious or honest but curious cloud service is a must, but we still need to protect the data privacy while achieving accurate performance. Our third approach aims to conditionally generate data streams in private multi-sensor scenarios, effectively distinguishing non-private applications while preserving the confidentiality of private ones.

## 1.1 Thesis Statement

Collaborative training with IoT sensor data from different users and sensors enables accurate models, but introduces data privacy concerns. Applying privacy-preserving techniques such as differential privacy can ensure data privacy, however, can also dramatically degrade the accuracy of IoT applications. By replacing a central server with a decentralized federated learning (DFL) framework with personalized reinforcement learning, augmenting the federated learning framework with a weight-enhanced local differential privacy (LDP) approach with a dynamic layer sharing mechanism, and using a contrastive learning based diffusion model for conditional synthetic data generation, we can achieve privacy-first smart IoT applications while ensuring their accurate performance for multi-user and multi-sensor scenarios.

## 1.2 Contributions of this Dissertation

In this thesis, we attempt to address the trade off between IoT data utility and privacy in **multi-users: residential users and global users** and **multi-sensors** scenarios and produce the following contributions:

- We propose *PFDR*, we propose a framework for **residential-users** where residents can share



their energy management plans while achieving a privacy-preserved and cloud-service-free residential energy management system (EMS) with a personalized federated deep reinforcement learning (PFDRL) framework. Instead of sharing just load forecasting results to identify standby energy, our framework can also allow residents to share their energy management plans to collaboratively train the reinforcement learning agent. Our design is based on the uniqueness of energy consumption in the residential area. (1) The energy consumption for each household in the residential area tends to share a similar energy usage pattern since they are in the same time zone and under the same weather conditions [5], [6], which reduces the training noise from outside households. (2) Residential area is a neighborhood with a limited number of households, which is a more controllable and reliable environment compared to global participants. This innate condition can provide collaborative training while omitting the need for cloud service by broadcasting the training gradient to other households. (3) Energy management task is not time-sensitive since the energy management plan can be updated once per several days so that the communication efficiency does not become a hindrance to our design.

- We propose `Atlas`, a private and accurate personalized federated local differential privacy (LDP) framework for IoT applications. We first design a layer-sharing mechanism called layer importance mask to separate the local model into global and personalized layers. Second, we design a weighted LDP mechanism and add noise to the global layers before transmitting them to the federated learning framework for aggregation. Third, we combine local personalized layers and aggregated global layers to perform IoT tasks.
- We propose `PrivateHub`, a novel synthetic datastreams generation method that utilizes contrastive learning in the diffusion model to conditional generating datastreams within the realm of private multi-sensor scenarios, ensuring accurate identification of non-private applications while simultaneously maintaining the concealment of private applications. `PrivateHub` is driven by the objective of harnessing the potent generative capacity of diffusion models to discern the aforementioned data types. Specifically, we propose a comprehensive framework for enhancing the performance of diffusion models in generating task-specific data, comprising two key stages: App-Conditioned Pre-training (ACP) and APP-Aware Fine-tuning (AAF). Initially, ACP involves pre-training a diffusion model using standard procedures on multi-sensor datastreams, enabling the model to generate app-conditioned data. Subsequently, AAF facilitates the diffusion model in further discriminating between privacy-sensitive and non-privacy-sensitive data through con-

trastive learning optimization. In this stage, an off-the-shelf application classifier is employed to extract features from various generated data, encouraging the model to produce data exhibiting feature distances closer to those of privacy non-sensitive ones.

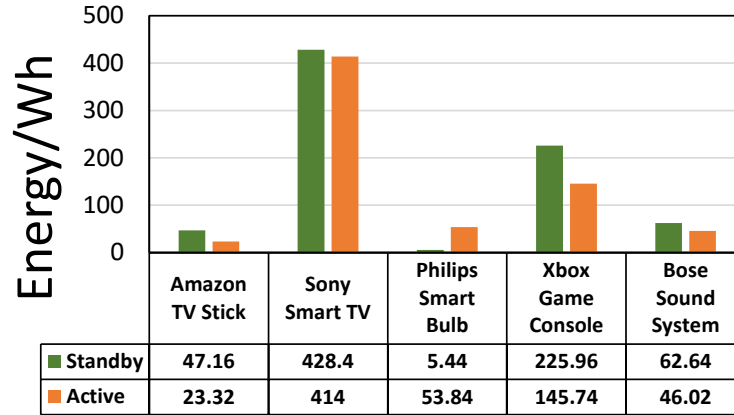
## Chapter 2

### **PFDRL: PERSONALIZED FEDERATED DEEP REINFORCEMENT LEARNING FOR RESIDENTIAL ENERGY MANAGEMENT**

Over the past few years, an increasing number of IoT devices have been installed in residential and commercial buildings to satisfy the increased demand for users' comfort, well-being, and quality of life. However, this increase in buildings' convenience, functionality, and connectivity require producing more energy which raises monetary costs and harmful gas emissions that directly cause environmental problems (e.g., climate change, water pollution, air pollution, and thermal pollution). In 2019, residential and commercial buildings were responsible for 40% of U.S. carbon emissions [7].

Standby energy is one of the reasons that cause the increasing energy consumption. By 2019, standby energy represents approximately 10% of residential electricity use in most developed countries and a rising fraction in developing countries [10]. Figure 2.1 shows an example of the particular energy consumption of 5 devices in 24 hours, which illustrates that standby energy consumption can be relatively large for certain smart devices if the devices are in standby mode for a period of time. Previous work [9], [11] has investigated the major causes of increased standby energy consumption of smart devices, and for the most part, energy is just wasted by devices waiting for commands. Despite the user comfort and efficiency that IoT devices provide to residences, the growing number of new IoT devices is having a negative effect on standby energy consumption. The sustained growth of smart home devices is expected to continue with a compound annual growth rate of 16.9% over the 2019-2023 forecast period [12], which brings a significant standby energy consumption problem.

Reducing the standby energy in buildings, especially in residential buildings has become an



**Figure 2.1: Results on daily energy consumption of 5 devices in a day. We calculate the standby energy usage and active usage following the setup: Amazon TV Stick, Sony Smart TV, Smart Bulb, and Bose sound system are active for 8 hours and standby for 16 hours, Xbox is active for 3 hours and standby for 21 hours [8], [9].**

important task for each residence. To achieve standby energy reduction in residential buildings, it is essential to accurately distinguish device standby energy usage from regular usage [13]. As a single user may not have enough data to train such a model, residents can collaboratively train their collected energy data to achieve better performance in device standby energy identification [14]. Traditional approaches require collecting device-level energy data for load forecasting using machine learning and deep learning methods to identify standby energy, then making energy management plans based on the prediction result [15], [16]. This prediction usually happens in the cloud. Although cloud platforms can bring high computational power and aggregate data from different users, they introduce the critical risk of potential personal data leakage [1], such as the central server becoming a malicious party. Federated learning (FL) is an emerging tool to address privacy issues, which allows IoT devices to collaboratively train a model. Existing approaches [17], [18] present a similar framework for load forecasting and energy management. However, such a framework still faces three problems: (1) Even though the data is stored locally, the FL framework still needs cloud support to aggregate a global model. Such a model contains all the training information from different local IoT devices, which is still vulnerable to training data recreation attacks by model inversion and still faces the potential risk of data leakage. The monetary cost of cloud service can also weaken the motivation of users to participate in the energy management system. (2) Localized energy management systems (EMS) require a long time to converge. Since only the load forecasting part is

done through FL, local EMS still lacks training examples, so achieving the best energy management performance is difficult. (3) Existing FL paradigms learn a single global model for every user, which can only improve the average accuracy, not the accuracy for each individual user. The diversity of users' data can lead to model convergence delay and cause low EMS performance for certain residents.

In this paper, we propose a framework where residents can share their energy management plans while achieving a privacy-preserved and cloud-service-free residential energy management system (EMS) with a personalized federated deep reinforcement learning (PFDRL) framework. Instead of sharing just load forecasting results to identify standby energy, our framework can also allow residents to share their energy management plans to collaboratively train the reinforcement learning agent. Our design is based on the uniqueness of energy consumption in the residential area. (1) The energy consumption for each household in the residential area tends to share a similar energy usage pattern since they are in the same time zone and under the same weather conditions [5], [6], which reduces the training noise from outside households. (2) Residential area is a neighborhood with a limited number of households, which is a more controllable and reliable environment compared to global participants. This innate condition can provide collaborative training while omitting the need for cloud service by broadcasting the training gradient to other households. (3) Energy management task is not time-sensitive since the energy management plan can be updated once per several days so that the communication efficiency does not become a hindrance to our design.

First, we introduce a decentralized federated learning (DFL) framework to enable distributed edge devices in the residential area to collaboratively train a model without a cloud server. The training parameters from each local model are broadcasted and aggregated between the smart home agents owned by each residence at a certain frequency, which removes the need of using a central cloud server and reduces the possibility of data leakage (e.g., malicious cloud server) and monetary cost from cloud service.

Second, to further improve the time to achieve the best EMS performance, we applied deep reinforcement learning (DRL) with DFL to share the EMS plan to reduce standby energy. We also use the load forecasting result as an input feature in the DRL framework to help with decisive action.

Third, we introduce personalized federated DRL (PFDRL) to maximize the EMS performance for each client in the sharing system as well as speed up the model converging time. We divide the network inside the DRL into base layers and personalized layers. Our training algorithm comprises

the base layers being trained in a federated fashion and personalization layers being trained only from local data with stochastic parameter descent. We evaluate the proposed PFDRl on the real-world Pecan Street dataset [19] and compare our results with four different types of EMS. Experimental results show that the proposed framework can achieve 92% load forecasting accuracy and save 98% of total standby energy consumption in a day for each residence in a residential area which outperforms all other methods.

## **2.1 Related Work**

### **2.1.1 Load Forecasting & Energy Management**

Various works in load forecasting majorly focus on two different aspects: aggregated household-level forecasting or device-level forecasting. Due to the fact that device-level forecasting suffers from high volatility and uncertainty of device usage that is not significant on aggregated loads, their major approaches are different. Kong et al. [20] introduced a centralized density-based clustering technique to evaluate and compare the inconsistency between the aggregated load and individual loads, then they adopted Long Short-Term Memory (LSTM) and designed a load forecasting framework for individual residential households. However, they require all data to be transmitted to a central hub, which can cause privacy concerns regarding sensitive user data. Din et al. [21] introduced a single device level load forecasting based on the Deep Neural Network approach, which they do not require the collection of lengthy historical data. However, their approach failed to conduct any load control and data privacy, which is a key contributor to our research.

### **2.1.2 Energy Data Privacy & Federated Learning**

Only a few works in home energy management systems have considered data privacy [22]–[24]. Most of their work aims to deal with the tradeoffs between energy data privacy and energy costs. For example, Yang et al. [23], developed an online control algorithm using the Lyapunov optimization technique to balance the problem between cutting down electricity bills and keeping the privacy of load requirements and electricity bill processes. A recent trend in maintaining user privacy is using federated learning approaches, where each agent processes its own collected data locally, and only transmits the calculated parameter to a central node. This reduces the amount of data transmission

and preserves user privacy. There have been several papers using federated learning. Wang et al. [25] train the machine learning model by distributing data across multiple edge nodes instead of sending them to a central node. Their proposed approach adaptively finds the best trade-off between local parameter update and global parameter aggregation under a given resource budget. Lee et al. [26] propose a novel federated reinforcement learning (FRL) approach for the energy management of multiple smart homes with home appliances, a solar photovoltaic system, and an energy storage system. However, the current FL focus on building a global model instead of a personalized model to maximize the energy management performance for each client. On the other hand, FL still utilizes cloud service to aggregate the model, which can be malicious and vulnerable and lead to personal data leakage from model inversion.

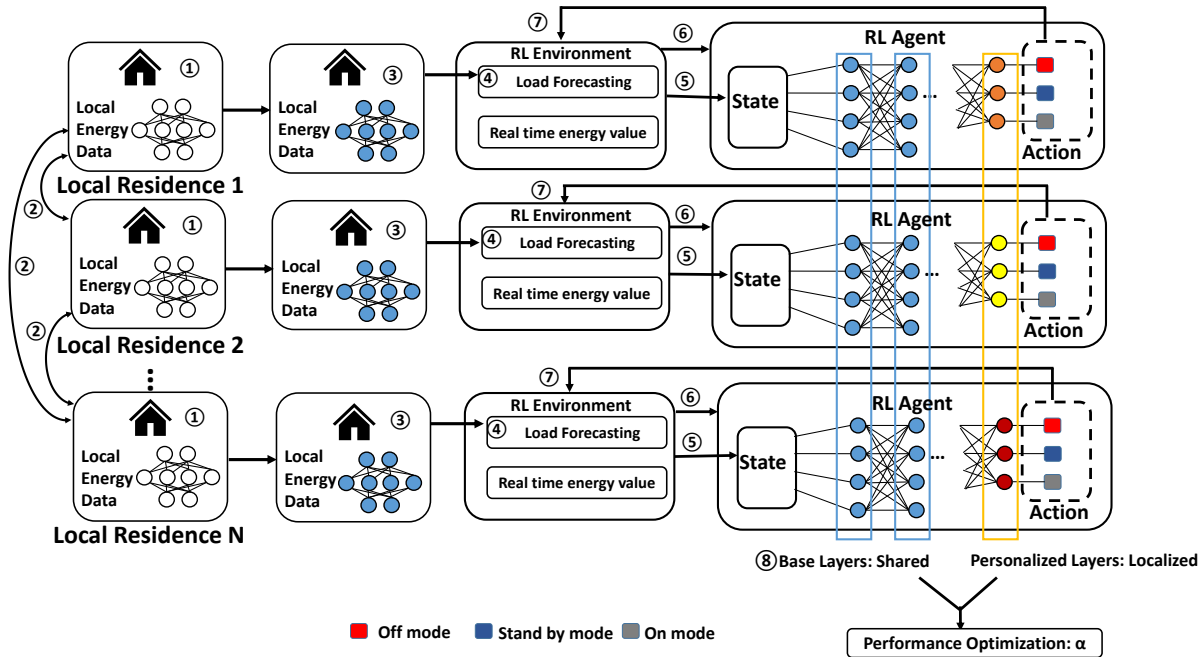
### 2.1.3 Energy Management Personalization

To better serve a large number of users with custom settings, personalized federated learning is proposed. In the FL approaches proposed in [27]–[29], the lower layers between all users are shared to the cloud server while keeping several user-specific upper layers on the edge. In this design, the more general features are saved on the lower layers while the upper layers capture a higher level of abstraction, which contains more user-specific features. However, these personalized methods rely on a pre-defined structure of model sharing, thus limiting the optimization of performance for each user. In this paper, we added a parameter  $\alpha$  that determines the number of layers to be transmitted, thus can optimize the performance for every single user. Other personalization energy management systems for residential buildings focus on personalized scheduling and demand response, with many using Bayesian networks for decision making [30], [31]. However, these methods rely on the full support of the cloud service, which might be malicious and bring data privacy issues.

## 2.2 System Design

### 2.2.1 System Overview

We present a personalized federated deep reinforcement learning (PFDRL) framework, which contains two parts: a decentralized federated learning (DFL) framework and a personalized deep federated reinforcement (PRL) framework. Figure 2.2 shows the structure of the proposed system.



**Figure 2.2: Overview of the PFDRL Framework:** ① Local energy data collection and training for standby energy identification; ② Parameters broadcasting at certain time frequency  $\beta$  between each residence; ③ Local testing with updated model; ④ Feed load forecasting result together with real-time energy value as deep reinforcement learning environment; ⑤ State observation; ⑥ Reward calculation; ⑦ Action selection to determine device mode; ⑧ Divide neural networks inside the deep reinforcement learning models as base and personalization layers using optimization parameter  $\alpha$ . Broadcast base layers at a certain time frequency  $\gamma$  and keep the personalization layers locally. Each residence will use the updated PFDRL framework to perform energy management locally.



Our proposed DFL framework removes the requirement for cloud services. First, each agent collects the device load data from every IoT device deployed in its local residence and trains a load forecasting model with the same machine learning method. Second, the agents broadcast the parameters of each device at a certain time frequency  $\beta$ , so that each agent has the parameter information from the same kind of device in other residences. Third, the load forecasting result is calculated based on the updated model. After each agent updates its local model by aggregating the qualified parameter for each device, the agent predicts the future power draw for each device to identify standby energy. In the meantime, the devices are still recording load data for the next training phase, and this process happens at the same interval, by default hourly. Fourth, the output of DFL, which is the load forecasting result, will be used as the input for the PFL framework along with the real-time load data. The proposed PRL framework will optimize the percentage of the base layer and personalized layer to achieve the best performance for each client in the system by the performance optimization parameter  $\alpha$ . Then the base layer will be broadcast to other agents in the system. The PFL framework is deployed on each agent in each resident. The load forecasting result can reflect the predicted mode for each device; the real-time load data can reflect the current mode for each device. Such information will be fed to the DRL agent to take the standby energy-reducing actions, such as switching the IoT device mode from standby to off.

### 2.2.2 Decentralized Federated Learning for Load Forecasting

Traditional distributed machine learning techniques require a certain amount of private data to be aggregated and analyzed at central servers (e.g., cloud servers) during the model training phase using distributed stochastic parameter descent (DSGD). Such a training process suffers from potential private data leakage risks. To address such privacy challenges, a collaboratively distributed machine learning paradigm, called federated learning (FL), was proposed for edge devices to train a global model while keeping the training datasets local and without sharing raw training data. However, traditional FL requires a cloud aggregator to obtain the global model by aggregating sparse parameters and sending this global model to the local agent. Such a method will create a global model in the cloud which suffers from not only potential private data leakage risks but expensive communication costs caused by the cloud.

In this paper, we focus on solving the above issue in a residential building. We remove the need for a central server by allowing the residents to process all collected data locally on edge

and broadcasting the model updates between the smart home agent in each residence inside the residential building. Such setup has the following benefits: (1) Remove the need for cloud service, which will save extra monetary cost from cloud usage, and avoid the possibility of a malicious central server. (2) The model information will only be broadcasted inside the residential building, which we believe is much easier to operate and privacy-preserved for residences.

**Locally Training Process:** In our system, we consider a residential home that includes  $N$  residents. Each residence  $n$  has an agent such as Google Home or Amazon Alexa, which has the connection between the IoT devices in certain residence  $n$ , where  $n \in \{1, 2, \dots, N\}$ . We denote  $A_n$  as the agent in the system, which represents its residence  $n$ . For each agent  $A_n$ , we have the same default training model initially, such as Long Short Term Memory (LSTM). We denote  $D_{Xn}$  as different IoT devices, in different residents, and  $X$  refers to the type of certain IoT device. Since each IoT device,  $D_{Xn}$  has its own local dataset (i.e., sensing time-series data from IoT nodes), we train the model separately for each device on the connected agent. For example, the TV in residence one, we define as  $D_{TV1}$ , TV in residence two, we define as  $D_{TV2}$ , the lighting in residence one, we define as  $D_{light1}$ , etc. We train the model for each device in each residence on the connected agent locally and separately. Thus every device has a certain parameter. For example, a parameter  $G_{TV1}$  is calculated for the TV in residence one in a certain time period. In each resident's home, an agent will record the parameter for all the resident's devices. As an instance, for residence one, the agent  $A_1$  has the parameter information  $G_{TV1}$ ,  $G_{light1}$ ,  $G_{HVAC1}$  and all the other devices that are connected to  $A_1$ .

Given a local dataset recorded by an IoT device  $D_{Xn}$ , our goal is to predict the future energy consumption for this certain device for the following hour. An LSTM model is used in the training set to learn the usage pattern of the device, and the testing set is used to predict the estimated energy consumption after the parameter aggregation. For each device  $D_{Xn}$ , we first predict the energy consumption  $V_{Xn}$  in every minute for the next hour. Then, we calculate and record the parameter for each device  $D_{Xn}$  locally, respectively, for broadcasting. To avoid high frequency broadcasting, we set  $\beta$  hours as the parameters broadcast rate to reduce the frequency of broadcasting parameters. In our experiment, we will determine the hyperparameter  $\beta$  that has the best result.

**Parameter Aggregation:** After each agent  $A_n$  determine the selected parameters from other agents for each device  $D_{Xn}$ , each agent will update the model with such parameters  $G_{Xn}$ . To do so,

we use DSGD for iterative updates, and the loss function to be optimized is defined as follows:

$$F(w) = \frac{1}{D_{X_n}} \sum_{n \in D_{X_n}} f(n, w) \quad (2.1)$$

where  $F(w)$  is the loss function for the updated model,  $f(n, w)$  is the loss function for the previous model, and  $w$  is the model's weight.

In the parameter aggregate phase, the agents obtain an updated model  $w_{t+1}$  for the next iteration as follows:

$$w_{t+1} = w_t - \eta \frac{1}{Nb} \sum_{n=1}^N \sum_{x \in B_{n,k}} \nabla f(x, w_t) \quad (2.2)$$

where  $\eta$  is the learning rate,  $B_n, k$  is the data sample for the  $k_t$ th round of training, and each local dataset size of  $b$ . All the collaborated agents repeat the above process until the model reaches convergence. Then we use the updated model to predict the energy consumption for a certain device for the next hour.

Algorithm 1 shows the training process of DFL load forecasting. Each home agent initialized its load forecasting model with the same structure for each device. The parameter for each model is set as random. For each device  $D_{X_n}$ , first, it locally trains its own model and finds the convergence  $W_{n,t}$ . After all models are converged, the smart home agent will broadcast the fine-tuned model parameters to all other smart agents from other residences. The broadcast frequency  $\beta$  will be determined from our experiment. Upon receiving all  $W_s$ , each agent aggregate the model parameters locally and update its own model. The load forecasting result for standby energy identification is also predicted using the updated model locally.

### 2.2.3 Personalized Federated DRL For Energy Management

After the load forecasting is made from the DFL framework, each DRL agent  $A_n$  needs to decide whether the mode of a certain device  $D_{X_n}$  should be changed or not. In the DRL framework, the action is made every minute based on the result from the DFL framework. So, the DRL agent performance is highly influenced by the DFL load forecasting accuracy. In our system, we first formulate this problem as a Markov Decision Process (MDP), and use a reinforcement learning (RL)

---

**Algorithm 1:** DFL load forecasting Algorithm
 

---

```

1 Initialize load forecasting model for each device  $D_{X_n}$ ;
2 Initialize model weight  $W_{n,0}$  at random;
3 for  $n=0$  to  $N$  do
4   for  $t=0$  to  $T$  do
5      $W_{n,t} \leftarrow SGD(W_{n,t-1}, \eta); (LocalTrainingStep)$ 
6     Check if each device  $D_{X_n}$  finished local training;
7     Broadcast  $W_{n,t}$  to other residences;
8     Receive  $W_s$  from all other residences;
9      $W_{n,t+1} \leftarrow \sum_{n=1}^N \frac{W_{n,t}}{N}$ 
10  end
11  Update local model with  $W_{n,t+1}$ ;
12  Localized load forecasting
13 end

```

---

method based on Deep Q-Network (DQN) as our base energy management system. Then, we divide the neural network inside the DRL as base layers and personalization layers. We broadcast the base layers to other residences and keep the personalization layers locally. The combined model with global aggregated base layers and localized personalization layers can achieve a collaborative training process as well as better energy management results for each residence.

### 2.2.3 DRL for energy management:

Reinforcement learning is a method where at each time step  $t$ , an agent observes a state  $s_t$  in an environment, takes an action  $a_t$  based on the observation, and receives a positive or negative reward  $r_t$  for the action taken. The objective of the DRL agent is to find an action policy  $\pi$  that would maximize the expected cumulative reward  $[\sum_{t=0}^{\infty} r_t]$ .

We first formulate this problem as a Markov Decision Process (MDP), denoted by  $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ , where  $\mathcal{S}$  is the state,  $\mathcal{A}$  is the action,  $\mathcal{P}$  is the probability between each two states and  $\mathcal{R}$  is the reward. Below, we introduce the elements in the MDP for our problem.

**State Space:** The state space  $\mathcal{S}$  is defined as the input of the DRL model. For each device  $D_{X_n}$ , it has three operation modes: off, standby, and on. Each mode can be reflected by energy consumption. In our system, the state space consists of two separate parts: The first part is the predicted energy consumption, which reflects the predicted device mode. The second part is the

real-time energy consumption, which reflects the real device mode.

We denote the predicted energy consumption by:

$$\mathcal{V} = \{V_1, V_2, \dots, V_t, \dots, V_T\} \quad (2.3)$$

where  $V_t$  means the  $t^{th}$  predicted energy consumption and  $T$  is the total number of predicted energy consumption. Since in the DFL prediction phase, each prediction is made for the next hour, the total number  $T$  is set as 60 minutes. The real-time energy consumption is recorded at the same timestamp as the predicted value:

$$\mathcal{RV} = \{RV_1, RV_2, \dots, RV_t, \dots, RV_T\} \quad (2.4)$$

where the value of  $t$  for  $RV_T$  is the same as  $V_t$ .

For each device  $D_{X_n}$ , it has a default value for each operation mode where  $V_{off}$  means the device is off,  $V_s$  means the device is in standby mode,  $V_{on}$  means the device is in on mode. For each  $RV_T$  and  $V_t$ , if the value is 0, we define the predicted and real mode of a certain device as off mode. If the value is between  $0.9 * V_s$  and  $1.1 * V_s$ , we define the predicted and real mode of a certain device as in standby mode. If the value is between  $0.9 * V_{on}$  and  $1.1 * V_{on}$ , we define the predicted and real mode of a certain device as on mode [32]. We denote  $S$  as predicted mode for certain device, where  $S \in \{S_{off}, S_s, S_{on}\}$  as off, standby and on mode. We denote  $RS$  as the real mode for certain device, where  $RS \in \{RS_{off}, RS_s, RS_{on}\}$  as off, standby and on mode.

**Action Space:** The action space  $\mathcal{A}$  is defined as the agent  $A_n$  can make decisions on whether the mode of a certain device  $D_{X_n}$  should be changed or not at time  $t$ . After receiving the predicted and real mode of certain devices, the action  $a_t$  is expressed in the following:

$$a_t = \begin{cases} 0, & \text{Off mode} \\ 1, & \text{Standby mode} \\ 2, & \text{On mode} \end{cases} \quad (2.5)$$

**Probability:** The state space is changed with certainty, so the probability between states is always 1.

**Reward:** Based on action space  $\mathcal{A}$ , the DRL agent receives a reward  $r(t)$  for each action taken at time  $t$ . In the system, the agent can receive positive rewards in one scenario: the predicted mode

**Table 2.1: Reward Function**

Ground truth mode	DRL action	Reward Value
On	On	10
On	Standby	-10
On	Off	-30
Standby	On	-10
Standby	Standby	10
Standby	Off	30
Off	On	-30
Off	Standby	-10
Off	Off	10

and the real mode for a certain device are the same at the time. In this case, the reward is set as 10. On the other hand, the agent can receive negative rewards in two scenarios: first, the predicted mode and the real mode for a certain device are different at time  $t$ , and the predicted mode should be moved up or down for one mode. In this case, the reward is set as -10. Second, the predicted mode and the real mode for a certain device are different at time  $t$ , and the predicted mode should be moved up or down for two modes. In this case, the reward is set as -30. An exception for the reward is that we wish once the real mode is standby, we want to change it to off mode, so we mark the reward as 30 in this scenario. Based on the aforementioned statement, we define reward  $r_t$  at time  $t$  for action  $a_t$  as shown in table 2.1.

**Q-value calculation:** The agent takes the action  $a_t$  that satisfies  $\max Q(s_t, a_t)$  and receives a reward  $r(s_t, a_t)$ . The goal is to find a policy  $\pi$  that will approximate the optimal Q-function  $Q^*(s_t, a_t)$  which always satisfies Bellman's optimality equation. That is,

$$Q^*(s_t, a_t) = r(s_t, a_t) + \kappa \cdot \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \quad (2.6)$$

where,  $Q^*(s_{t+1}, a_{t+1})$  is the next-step's optimal Q value.

As there will also be new load data coming from each device, we keep training our DRL agent to achieve the best performance over time. After the first time DRL agent training process, each DRL agent has its own DRL model ready for broadcast. We apply a performance optimization parameter  $\alpha$  to determine which part of the model will be broadcasted and updated to achieve the best energy management performance in the following section.

### 2.2.3 Federated Personalization for Reinforcement Learning:

We divide the neural network in DRL into two parts: base layers and personalization layers. Base layers act as the shared layers which are trained in a collaborative manner. Equation 2.7 shows the model aggregation step for base layers:

$$W(DRL_B)_{n,t+1} = W(DRL_B)_{n,t} - \delta \sum_{n=1}^N \sum_{x \in B_{n,k}} \frac{\nabla f(x, W(DRL_B)_{n,t})}{N} \quad (2.7)$$

where  $W(DRL_B)_{n,t}$  denotes the base layers model weight for the  $n^{th}$  residence at time  $t$ .  $\delta$  is the learning rate,  $B_n, k$  is the data sample for the  $k_t$ th round of training, and each local dataset size of  $b$ . By uploading and aggregating only part of the models, PFDRL requires less computation and communication overhead, which is essential in IoT environments. On the other hand, in PFDRL, the participants share the parameters of their DRL model, which can achieve better performance in a shorter time because of the collaborative training.

While the personalization layers are trained locally, thereby enabling to capture of personal information of IoT devices. Equation 2.8 shows the model aggregation step for PFDRL:

$$W(PFDRL)_{n,t+1} = W(DRL_P)_{n,t} + W(DRL_B)_{n,t+1} \quad (2.8)$$

where  $W(DRL_P)_{n,t}$  is the weight of personalization layers, and  $W(PFDRL)_{n,t+1}$  is the updated model weight for PFDRL. In this way, the globally-shared base layers can be broadcasted to participating IoT devices for constituting their own EMS plan with their unique personalization layers. Thus, our proposed PFDRL is able to capture the fine-grained information on a particular device for superior personalized inference or classification and address the statistical heterogeneity to some extent. We define an 8 hidden layers architecture of neural networks inside the DRL agent and determine a performance optimization parameter  $\alpha$  to decide the proportion of base layers and personalization layers. To avoid high-frequency broadcasting of base layers, we set  $\gamma$  hours as the parameters broadcast rate to reduce the frequency of broadcasting parameters. In our experiment, we determine the hyperparameter  $\gamma$  that has the best result.

Algorithm 2 shows the training process of personalized deep federated reinforcement (PFDRL). DRL agents will take actions based on DRL state space, which is the load forecasting value and

real-time energy value. The reward will be calculated based on the action. The system calculates the  $Q$ -value and finds the maximum value based on the selection of action space. We adopt the Huber Loss function [33] which acts quadratic for small errors and linear for large errors. This prevents the network from having a dramatic change while processing outliers. After the local training step is done, we select  $\alpha$  base layers of the model broadcasted to other residences, each residence combines the updated model with the aggregate base layers and localized personalization layers to perform energy management plans.

**Evaluation:** In this section, we evaluate our proposed framework with different hyperparameter settings. We also evaluate our proposed framework comparing with different sets of compared methods in load forecasting performance and energy management performance.

## 2.2.4 Datasets and Experiment Settings

**Energy Consumption:** The proposed framework is applied to a real-world dataset for performance demonstration: the Pecan Street dataset [19]. The Pecan Street dataset contains the energy consumption for every minute of different home appliances such as bedrooms lights, dryers, HVAC, etc in 669 residents in Texas from 2013-01-01 to 2017-12-31. Since the recorded data has some missing data points, also the devices in each residence are not exactly the same, we select 200 residents that have the HVAC, electric furnace, dishwasher, dryer, refrigerator and oven load records as our experiment dataset since they are considered to cost the most of standby energy in a household [34].

**Electricity Price:** Since the electricity price can be divided into fixed-rate electricity plans and variable rate electricity plans. We obtain the electricity price from the websites of Texas Electricity Rates [35] and the websites of Energy Information Administration [36]. For fixed-rate electricity prices in TX, the average rate is 11.67 cents per kilowatt-hour (kWh). For variable rate electricity prices in TX, the range is between 0.08 cents to 20 cents per kWh depending on the time. We compare both in our experiments to see the price difference.

**Experiment Settings:** The experiments are deployed in our local server with a GTX 2070 super GPU. For both energy load forecasting and management, we first use 80% of the energy consumption dataset as the training set and use them to calculate the parameter and aggregated the parameters from other agents to get the updated model. Second, we use the rest 20% of the dataset for testing. For the hyperparameters in PFDRL, we set the learning rate as 0.001, discounted rate as 0.9, the memory capacity as 2000, and the target replace iteration as 100. Each hidden layer has 100



---

**Algorithm 2: PFDRL Algorithm**


---

```

1 Initialize DRL environment with load forecasting result  $\mathcal{V}$  and real-time energy value  $\mathcal{R}\mathcal{V}$ 
  for each device  $D_{X_n}$ ;
2 for  $n=0$  to  $N$  do
3   for  $t=0$  to  $T$  do
4      $a_t = \text{random}(0,2)$ ;
5     or,  $a_t =_a Q(s_t, a)$ ;
6     TakeAction( $a_t$ );
7      $A.append(a_j)$ ;
8      $S.append(s_j)$ ;
9      $a_t = A[t]$ ;
10     $s_t = S[t]$ ;
11    Compute  $r_t$  from  $s_t, a_t$ ;
12     $s_{t+1} = S[t + 1]$ ;
13    for each iteration do
14      Calculate Q value,  $y_i = r(s_i, a_i) + \kappa \cdot \max_{a'} Q(i+1, a')$ ;
15      Calculate error,  $\tau_i = y_i - Q(s_i, a_i)$ ;
16      Calculate loss,  $\mathcal{L} = \frac{1}{B} \sum_{e_t \in B} \mathcal{L}(\tau)$ ;
17      Update network parameters; (LocalTrainingStep)
18    end
19    for  $\alpha \in (1, 8)$  do
20      Broadcast  $\alpha$  layers in DRL to other residences;
21      Keep  $(8 - \alpha)$  layers locally;
22      Receive  $\alpha$  layers from all other residences;
23      Do Equation 2.7
24    end
25  end
26  Do Equation 2.8;
27   $\alpha = \alpha + 1$ 
28 end

```

---

Method	Load forecasting	EMS	Model in the local area	Data Privacy	Model training for small amount of data	Sharing EMS plan to achieve best performance in short time	Personalization
<b>Local</b>	Local NN	Local RL	✓	✓	✗	✗	✓
<b>Cloud</b>	Cloud NN	Local RL	✗	✗	✓	✗	✗
<b>FL</b>	Federated learning	Local RL	✗	✗	✓	✗	✗
<b>FRL</b>	Federated learning	Federated RL	✗	✗	✓	✓	✗
<b>PFDRL</b>	Decentralized federated learning	Personalized federated RL	✓	✓	✓	✓	✓

**Figure 2.3: Comparison methods.**

neurons followed by a ReLU function. The output layer has 3 neurons providing  $Q$ -values of the state of the three modes.

### 2.2.5 Compared Methods

For load forecasting, we compare four prediction algorithms: Linear regression (LR) [37], Support vector machine (SVM) [38], Back-propagation network (BP) [39], and Long short-term memory (LSTM) [40] with the same experiment settings and choose the method with the best performance applied to the PFDRL model.

To evaluate PFDRL, we choose the following four methods as comparisons.

- Local based load forecasting + local based EMS [41] (Local).
- Cloud based load forecasting + local based EMS [42] (Cloud).
- Federated learning based load forecasting + local based EMS [18] (FL).
- Federated learning based load forecasting + federated learning based EMS [18] (FRL).

We train the models with the same dataset under their settings. Figure 2.3 shows the details of each compared method.

## 2.2.6 Performance Metrics

1. *Hyperparameters selection.* We measure the hyperparameter  $\alpha$ ,  $\beta$ , and  $\gamma$  to determine the threshold for broadcast frequency and the number of broadcasted layers in our system.

2. *Prediction accuracy.* To measure the prediction accuracy of energy consumption, we measure the prediction accuracy as below:  $Ac_n = 1 - \frac{|V_n - RV_n|}{RV_n}$  where  $Ac_n$  is the prediction accuracy of  $n^{th}$  prediction,  $V_n$  is the predicted value of  $n^{th}$  prediction and  $RV_n$  is the real value of  $n^{th}$  prediction.

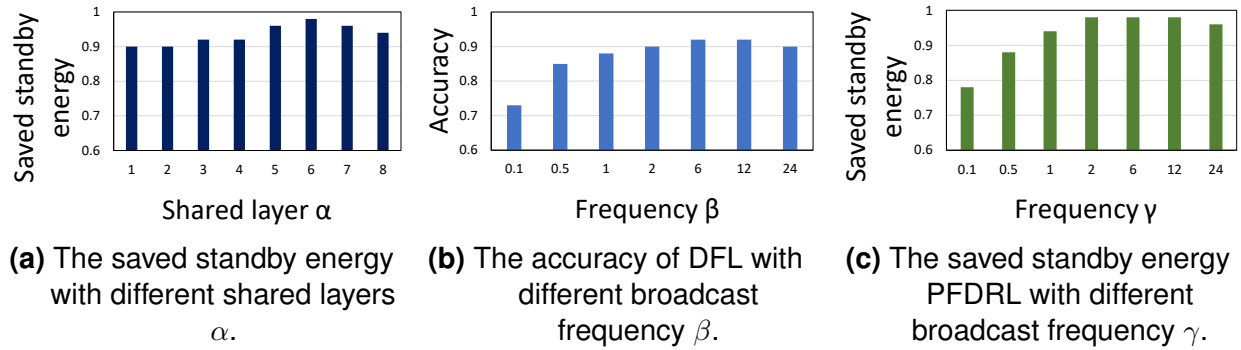
3. *Saved energy value.* To measure the saved energy value, we use  $RV_n - V_n$  to calculate the value.

4. *Saved monetary cost.* We calculate the total monetary cost based on real price dataset [35], [36] for fixed rate electricity plans and variable rate electricity plans under  $C_{D_{X_n},t} = (RV_{n,t} - V_{n,t}) \cdot p_t$  where  $C_{D_{X_n},t}$  is the monetary cost for device  $D_{X_n}$  at time  $t$ .  $p_t$  is the energy price at time  $t$ .

5. *Time overhead.* We use training time latency and testing time latency to show the time overhead of the load forecasting methods and the energy management methods.

## 2.3 Experimental Results

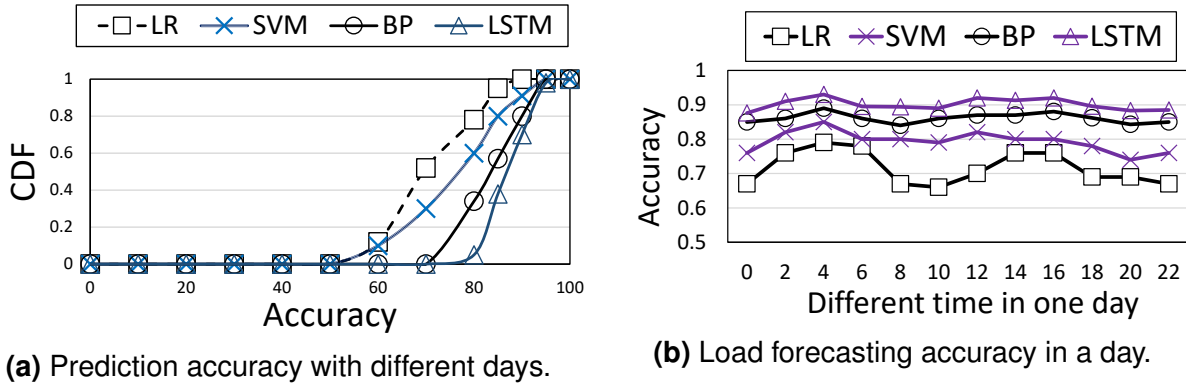
**Hyperparameters Selection of PFDRL:** Figure 2.4(a) shows the saved standby energy of the PFDRL framework with different shared layers  $\alpha$ . We use  $\alpha \in \{1, 2, 3, 4, 5, 6, 7, 8\}$  to determine the best  $\alpha$  of the proposed framework. We observe that  $\alpha = 6$  has the best result, which means the best performance comes from we set 6 layers as base layers and 2 layers as personalization layers. Figure 2.4(b) shows the accuracy of our proposed DFL framework with different broadcast frequencies  $\beta$ . We employ  $\beta \in \{0.1, 0.5, 1, 2, 6, 12, 24\}$  to adjust the best frequency of the proposed framework. We observe that  $\beta = 6$  and 12 have the best prediction accuracy for load forecasting, which means the parameters should be broadcasted every 6 or 12 hours. Since higher broadcast frequency will cause lower communication efficiency, we choose  $\beta = 12$  as the best frequency of our framework. Figure 2.4(c) shows the saved standby energy of our proposed PFDRL framework with different broadcast frequencies  $\gamma$ . We employ  $\gamma \in \{0.1, 0.5, 1, 2, 6, 12, 24\}$  to adjust the best frequency of the proposed framework. We observe that  $\gamma = 2, 6$ , and 12 have the best performance, which means the parameters should be broadcasted every 2, 6, or 12 hours. Due to the same reason as described in Figure 2.4(b), we choose  $\gamma = 12$  as the best frequency of our framework.



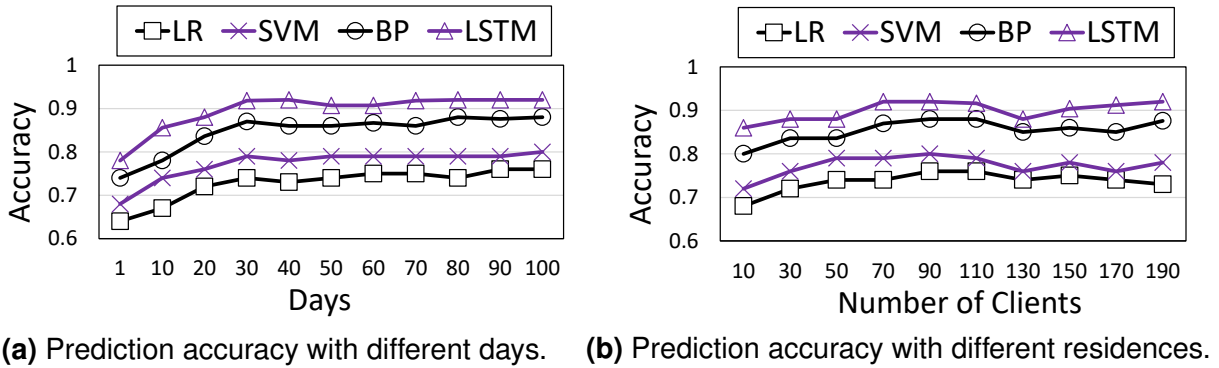
**Figure 2.4: System Parameters**

**Load forecasting Accuracy:** Figure 2.5(a) shows the cumulative distribution function (CDF) of the load forecasting result. The result follows  $LR < SVM < BP < LSTM$ . For LR, it's normal to face under-fitting and low precision, so the load forecasting accuracy is lower. For SVM, its performance with large datasets is lower than the others. For BP, it is easy to fall into a local extreme value, and the weights converge to a local minimum point, which causes the network training to fail. For LSTM, it can capture the long-term pattern based on the memory cell, which can bring higher load forecasting accuracy. Figure 2.5(b) shows the load forecasting accuracy in a day at different times. The result follows  $LR < SVM < BP < LSTM$  due to the same reason as explained in Figure 2.5(a). We also observe that the accuracy from 2 AM to 6 AM and from 12 PM to 16 PM are higher than the other time in the day. The reason is that, in such a time frame, residences usually have the same energy usage patterns for each device. From 8 AM to 10 AM and evening time, the energy usage in different residences is vary depending on the date.

Figure 2.6(a) shows the prediction accuracy while we accumulatively train the DFL framework with different numbers of days. We set the number of residences as 100 in this experiment. The result follows  $LR < SVM < BP < LSTM$  due to the same reason as explained in Figure 2.5(a). Since we accumulatively train the DFL framework, for each hour, each agent has the aggregated parameter for each device. The updated parameter will be used for the next training period, which will improve the prediction accuracy over time. On the other hand, from day 1 to day 30, the growth of accuracy is higher than from day 70 to 100. The reason is that the aggregated parameter tends to approach the best value for load forecasting. Figure 2.6(b) shows the prediction accuracy with different numbers of residences participating in the DFL framework. We set the number of days as 365 in this experiment. For the number of residences under 100, the result follows  $LR < SVM < BP < LSTM$



**Figure 2.5: Prediction Accuracy Comparison**



**Figure 2.6: Prediction Accuracy Comparison with Different Experiment Settings**

due to the same reason as explained in Figure 2.5(a). For the number of residences above 100, the result shows a drop. The reason is that the four methods use all the parameters or data to train the model together. Such methods can indeed improve the average accuracy when the number of total participants is small. When the number of residences goes up, the number of different kinds of load patterns also goes up. In this case, using all the parameters or data to train the model may cause prediction accuracy to drop in some devices.

**Performance Comparison with Compared Methods:** Figure 2.7 shows the amount of saved energy per residence and the percentage of standby energy usage versus different training days. The result for saved energy per client follows  $Cloud \approx FL \approx FRL < Local \approx PFDRL$ . Because the local method and PFDRL have personalization, so the EMS plan is more accurate, which leads to the result that more energy can be saved from standby energy. The result for achieving the best performance time follows:  $PFDRL \approx FRL < FL \approx Cloud < Local$ . Because PFDRL and FRL are

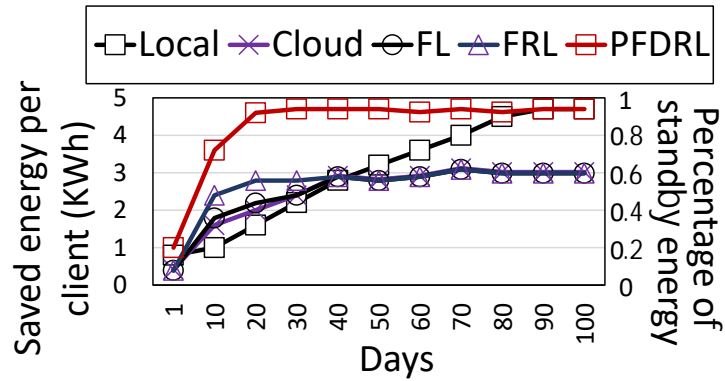


Figure 2.7: Saved energy per residence via days.

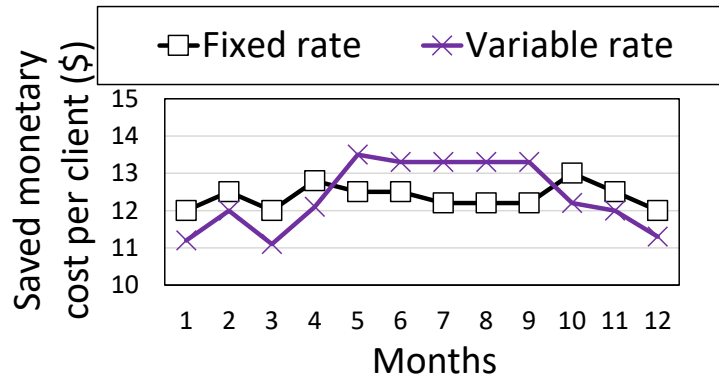
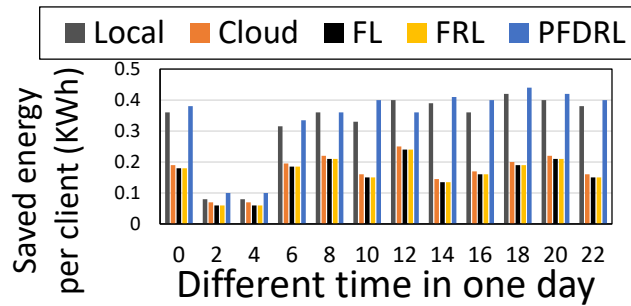
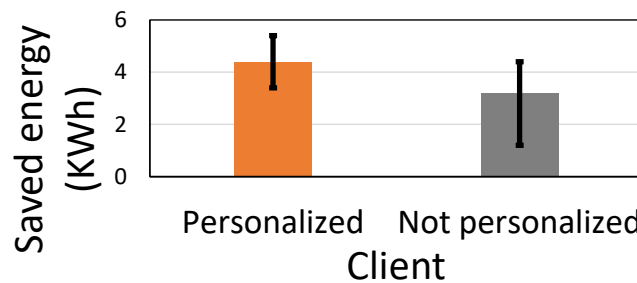


Figure 2.8: Saved monetary cost per residence via months.

sharing the EMS plan with other clients in the system, which leads to the fact that sharing the EMS plan with all participants can speed up the system to achieve better performance. For FL and cloud, since they only do load forecasting in the sharing mechanism, they have a more accurate input feature for the localized EMS, but since the EMS plans are not shared, so they will need to spend more time to achieve the best performance. For local based method, since both load forecasting and EMS plan are locally based, it has the lowest speed. Figure 2.8 shows the saved monetary cost per residence and the percentage of the total monetary cost versus different months. We compare both the fixed-rate electricity plan and the variable rate electricity plan using our system. The result follows Fixed Rate  $\approx$  Variable Rate. Since the amount of saved energy is the same, the difference in the total monetary cost in a month is between the electricity plan. On average, we can observe that Fixed Rate  $\approx$  Variable Rate. From April to June, the variable rate plan will save more money for each resident. From August to October, the fixed-rate plan will save more money for each resident.



**Figure 2.9: Saved energy per residence in a day.**

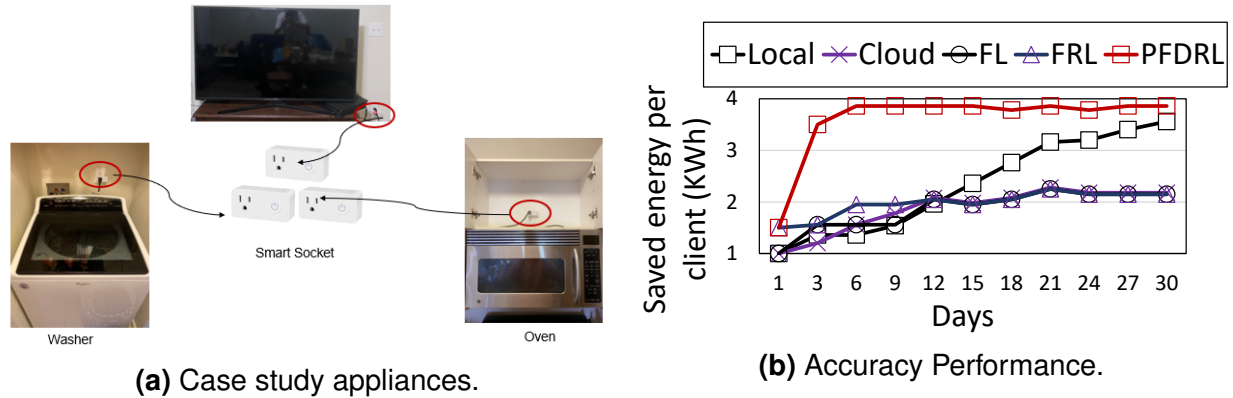


**Figure 2.10: System performance in personalization.**

Figure 2.9 shows the amount of saved energy per residence at different times of the day. We can observe that the result follows  $\text{Cloud} \approx \text{FL} \approx \text{FRL} < \text{Local} \approx \text{PFDRL}$  due to the same reason as explained in Figure 2.7. We can also observe that, between 2 AM and 4 AM, the saved energy is at a minimum. The reason is that the total usage of energy is at the lowest level. On the other hand, between 12 PM and 0 AM, the saved energy is at maximum since residents are using more energy at such time in the day. Figure 2.10 shows the system performance in personalization. We show the mean accuracy of the personalized model and not the personalized model. We can observe that the personalized model has better performance than the not personalized model. Also, from the error bar, we can observe that personalized model can achieve better performance for most residences.

## 2.4 Case study

We also test our proposed method in the case study. Figure 2.11(a) shows the appliances we used in this case study. We record the energy usage for a single washer, TV and oven for a month duration and use it as one test set to test our model. The purpose of this case study is to see that, how will the



**Figure 2.11: Case Study**

system perform for a newly added home in the system.

Figure 2.11(b) shows the case study performance. The result for saved energy per client follows:  $\text{Cloud} \approx \text{FL} \approx \text{FRL} < \text{Local} \approx \text{PFDRL}$ , on the other hand, the result for achieving best performance time follows:  $\text{PFDRL} \approx \text{FRL} < \text{FL} \approx \text{Cloud} < \text{Local}$  due to the same reason as shown in Figure 2.7. We can also see that it takes around 7 days for the newly added client to achieve best performance.

## 2.5 Conclusion

In this paper, we propose a privacy-preserved energy management system that can achieve the best performance in a short time and minimize the energy usage caused by standby energy. First, we introduce decentralized federated learning (DFL) framework to enable distributed edge devices to collaboratively train a model without using cloud service. Second, to further improve the time to achieve the best EMS performance, we applied deep reinforcement learning (DRL) with FL in order to share the EMS plan. Third, we design a personalized federated DRL (PFDRL) to maximize the EMS performance for each individual client in the sharing system by dividing the network inside the DRL into localized-based layers and personalized layers. We evaluate the proposed PFDRL energy management system on the real-world Pecan Street dataset, which saves 98% of total standby energy consumption per day in a residential building.



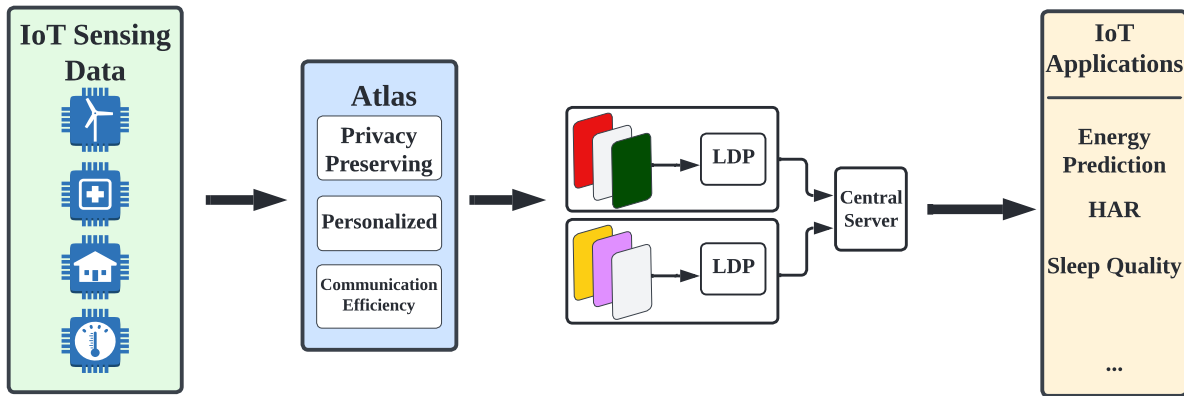
## Chapter 3

### **ATLAS: ENSURING ACCURACY FOR PRIVACY-PRESERVING IOT APPLICATIONS**

The Internet of Things (IoT) technologies enable a large number of applications that involve a rich set of sensors, such as energy consumption prediction [43], traffic forecasting [44], health monitoring [45], and human activity recognition (HAR) [46]. The key to enabling these IoT applications is machine learning or deep learning models trained on a massive amount of data. However, the edge devices of IoT applications often have access to only limited data, which is not sufficient for modern deep learning model training. The common solution is to aggregate all clients' data in a central server and train a global model using all available data, which raises problems with data privacy, sub-optimal accuracy performance and communication latency. Given the privacy-sensitive, data heterogeneity and latency-sensitive nature of IoT sensing data (e.g., medical data for health monitoring, activity data for HAR), we need to have a privacy-preserving, personalized and communication efficiency framework for smart and reliable IoT applications.

To address the intertwined challenges of data scarcity, data privacy, and communication latency in IoT applications, federated learning (FL) provides a popular method to achieve distributed machine learning among numerous devices without sharing raw data with a central server for collaborative training [47], FL has been widely applied in the IoT domain [48]. Most state-of-the-art FL frameworks share and aggregate the weights of the models trained on sensitive client data instead of the raw data directly [28], [47], [49], [50]. Unfortunately, FL still cannot guarantee sufficient privacy for sensitive data. Recent works have demonstrated that shared weights also can leak sensitive information by model inversion during the process of model updating [51], [52].

To further enhance privacy in FL training, recent studies applied more advanced privacy-persevering methods like differential privacy (DP) to FL models [53], [54]. Differential privacy adds



**Figure 3.1: High-level Atlas Architecture**

$\epsilon$  random noise [55] to provide strong guarantees of privacy for individuals within a dataset. Recent studies of FL models with (local) differential privacy limit the leakage of sensitive information in FL training [54] and avoid the possibility of information leakage during data transmission. But, the addition of random noise necessarily reduces the accuracy of these models. Yu et al. [56] confirmed FL models introduce a fundamental conflict between protecting privacy and higher model utility/performance. They further claimed that model personalization methods like local adoption are effective in improving FL model performance, even for models with privacy protection. In summary, a suitable FL system is in demand to guarantee the three problems for IoT applications. (1) **Privacy.** The LDP approaches should have a strong privacy guarantee for IoT applications. (2) **Model performance.** The design of the algorithm can work collaboratively with the DP mechanism to ensure comparable model accuracy to non-privacy-preserving methods. (3) **Communication efficiency.** The personalization algorithm needs to optimize the number of parameters transferred to the aggregation step resulting in lower communication costs for different IoT applications.

#### **Status Quo and their Limitations.**

(1) **Local differential privacy mechanisms.** As the practice of utilizing sensitive user data to train deep learning models has amplified privacy worries across various sectors, recent research [57] has shown that using FL alone can not meet the requirements of data privacy due to model inversion. Incorporating local differential privacy into federated learning serves as an effective method to ensure stringent privacy guarantees. However, existing LDP methods [58]–[60] add the same LDP noise to all the model layers for simplicity which is not practical, such methods ignore the fact

that the model weights from different layers vary significantly. Presuming that the weights across all layers fall within a constant range can lead to significant fluctuations in the estimated model weights, ultimately resulting in suboptimal model performance.

(2) **Federated learning frameworks.** There are many existing personalized FL models using different technologies like adding user context [61], transfer learning [61], [62], multi-task learning [63], meta-learning [64], and the others [56], [65]. All the sets of personalized methods aim to solve the problem of data heterogeneity and achieve higher performance for individual users. The common personalized FL can be divided into two main categories: (1) model enhancement based federated learning personalization models [56], [66], [67]. (2) transfer learning based federated learning personalization models [29], [61], [62], [64].

However, the existing federated personalization solutions are not the perfect partner for the differential privacy setting. For model enhancement based federated learning personalization models, such as ClusterFL [66], the main concept is to enhance model training accuracy by identifying data relationships across nodes by clustering the clients in the central server. It expedites convergence and maintains accuracy by efficiently removing slower or less correlated nodes within each cluster. Yet, such a method requires high-quality model information (e.g., model gradient) for clustering based centralized aggregation. Directly adding LDP noise such as the Gaussian mechanism to such models will cause dramatic performance reduction [68].

Transfer learning-based personalization methods aim to divide the model into two parts, the sharable layers for global aggregation and the personalized layers for local updates. Such a setup is more compatible with the FL-LDP setting since the differential privacy noises are added into each layer which does not require high-quality gradients to train the whole model. But existing transfer learning types of FL personalization methods, such as Wang et al. [62], FedPer [29], and Reptile [64], cannot optimize the model performance since their sharable layers are simply fixed. Practically, the layers' importance for global aggregation and local updates are dynamically changing over time. Models for each client do not know which part of the layers can contribute more to the global model and which part of the layers is more helpful to the local models for each iteration [69]. Thus, to further enhance the performance of FL-LDP setup, a reliable and personalized layer selection mechanism is in demand to customize the layer weights that need to be transferred to the global server. It can improve the model utility while reducing the communication cost compared to traditional FL approaches.

(3) **Uniqueness of IoT data.** While Federated Learning (FL) holds substantial importance in solving data privacy for IoT application collaborative training, the majority of FL research primarily utilizes well-established datasets like CIFAR-10, MNIST and CIFAR-100 for evaluation. However, these datasets don't reflect the authentic characteristics and complexities of real-world IoT data, which fails to prove their practicality of IoT applications.

**Overview of Proposed Approach.** Motivated by the limitations of existing solutions, we present *Atlas*, a privacy-first practical personalized federated learning framework with dynamic local differential privacy designed specifically to resolve the above challenges simultaneously. *Atlas* aims to provide a unified privacy-preserving FL framework for IoT applications to (1) collaborative training deep learning models with different clients with privacy guarantees, (2) learn a personalized model for each participating client with optimal performance under the privacy guarantee. (3) achieve reasonable communication costs for IoT applications.

Figure 3.1 shows the high-level system architecture of our proposed system. The core idea of *Atlas* can be split into two parts. First, we design a dynamic layer importance selection and sharing framework based on a self-learned layer importance mask mechanism for each client. The layer importance mask is refined along with the model parameters during the FL training process, which customizes the global and local layers according to the characteristics of each layer in every client. During the training process, the layer importance mask is randomly initialized and interacts with model layer weights to calculate the global importance of each layer during updating, the layers with higher importance scores will contribute to the global model training while the layers with lower importance scores will be pruned and contribute to local model training [70]. Note that since we only update selected layers to the central server, the communication cost can also be minimized during this process. Second, we add LDP to globally important layers. Instead of adding the same LDP noise to all selected model layers, we utilize the layer importance mask from the first step to add different noise for different selected layers and propose a more dynamic and practical LDP mechanism to demonstrate how the range influences the variance in the model weights.

**System Implementation and Performance,** we developed the *Atlas* system and performed extensive experiments using real IoT-related application data. The model performance is evaluated specifically on five IoT tasks, We compare *Atlas* with 8 popular baseline functions in the personalized FL and LDP domain and achieved comparable model performance with non-privacy-preserving methods with privacy guarantee.

To the best of our knowledge, `Atlas` is the first FL framework that optimizes the personalized model by dynamic layer selection during training while adding practical IDP to guarantee model utility, privacy, and communication efficiency at the same time in an IoT scenario. Compared to other state-of-the-art FL methods, `Atlas` demonstrates the personalization power of our proposed dynamic layer selection strategy with LDP, which effectively resolves the limitations of the existing approaches. More specifically, table 3.1 shows the comparison between `Atlas` and status quo methods. `Atlas` improves the model utility compared to LDP-FL by using a novel layer selection personalization method, while other personalized FL frameworks like FedMask, are hard to train along with differential privacy. Compared to FedPer and other local fine-tuning methods, our proposed global layer selection technique optimized the model structure for both global training and local fine-tuning. In addition, since we only update selected layer parameters to the central server in `Atlas`, the communication cost can be reduced compared to traditional FL frameworks like FedAvg and ClusterFL.

**Table 3.1: Comparing `Atlas` with existing FL approaches in IoT scenario.**

Approach	Personalization	Privacy-Preserving	Communication
FedAvg [47]	✗	✗	✗
ClusterFL [66]	✓	✗	✗
FedPer [29]	✓	✗	✓
FedDL [70]	✓	✗	✗
FedMask [67]	✓	✗	✓
LDP-FL [54]	✗	✓	✗
<code>Atlas</code>	✓	✓	✓

### 3.1 Background and motivation

In this section, we introduce the background and motivation of our work. We first show the need for applying privacy-preserving techniques to federated learning frameworks and the drawbacks of current methods. Then we discuss the background of current federated learning frameworks and the benefit of personalization, which we believe is the key to overcoming the drawbacks of directly

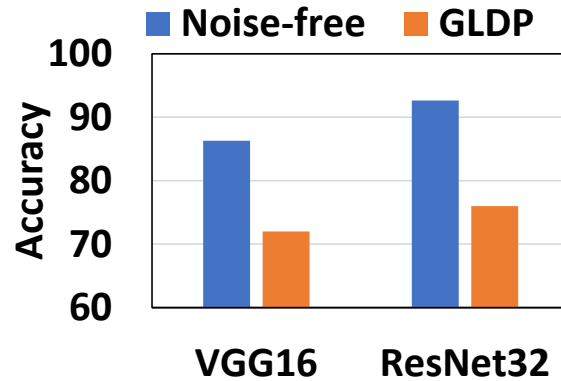
applying privacy-preserving techniques.

### 3.1.1 Privacy Preserving for FL

Federated learning (FL) enables collaborative training by using a central server to aggregate the locally trained model weights instead of sharing users' local data to maintain data privacy. However, recent research works [54], [71], [72] have shown that FL itself is not private enough. The FL framework still requires cloud support to aggregate a global model. Such a model contains all the training information from different users and remains vulnerable to training data attacks with malicious cloud server or model inversion which result in facing the potential risk of data leakage. One way to enhance the FL framework with privacy guarantee is to apply differential privacy or local differential privacy mechanisms. However, the trade-off between privacy guarantee and model accuracy is quite tricky for IoT applications. We conduct experiments to investigate how significantly the accuracy is dropped by adding Gaussian local differential privacy (GLDP) to FedAVG [47]. In this experiment, we apply noise-free FedAVG and GLDP FedAVG ( $\epsilon = 2$ ) and adopt the VGG16 [73] and ResNet32 [74] as the default model configurations. We train the models on the CIFAR-10 dataset (note that we use CIFAR-10 as an example for general purposes). As Figure 3.2 shows, adding GLDP to FedAVG causes a dramatic accuracy drop from 86.4% to 72.3% for VGG16, and from 92.3% to 76.1% for ResNet32. Such a performance drop for IoT applications is not acceptable in practice. Therefore we aim to enhance the model performance while ensuring the guarantee of data privacy.

### 3.1.2 The Need for Personalization in FL

Federated learning facilitates collaborative model training while ensuring that local data remains confidential and unshared. Among various FL strategies, FedAvg [47] stands out as a foundation method. In the FedAvg framework, a central server coordinates with numerous client devices. During each iteration of communication, a specific subset of these devices is chosen to participate. The chosen devices engage in training activities on their respective local data, employing a uniform learning rate and identical count of local epochs to cultivate their individual local models. These models undergo updates via stochastic gradient descent (SGD). Subsequently, the local model updates are transmitted to the central server, where they are collectively averaged, resulting in



**Figure 3.2: Comparison of Accuracy between Noise-free and GLDP FedAVG in Different Model Configurations.**

an update to the global model. This updated global model is then distributed back to all the devices. However, it is crucial to note that the data across devices typically follows a non-IID distribution in practical scenarios. This statistical heterogeneity poses challenges in developing a shared global model that can effectively generalize across all participating devices. Therefore, introducing personalization is crucial to effectively address the challenges arising from statistical heterogeneity.

## 3.2 Related Work

**Federated Learning and Personalization** Numerous studies from a wide range of research have explored the usage of FL methods from various perspectives. Some recent works in the FL domain have focused on the usage of IoT devices. Li et al., [75] presented an overview of challenges, open problems, and issues associated with FL by considering the heterogeneity of devices while assuming all clients are resource-boundless. Yang et al., [76] focused on the FL settings categorization, Niknam et al., [77] elaborated on the issues of FL setup in a wireless environment. Adapting a global model can help achieve personalization. Existing works in this domain mainly use two separate steps that result in extra overhead. With first having the global model learned in a federated fashion, then fine-tuning the global model for each client using its local data. The common personalized FL can be divided into two main categories: (1) model enhancement based federated learning personalization

models [56], [66], [67]. (2) transfer learning based federated learning personalization models [29], [61], [62], [64].

**Differential Privacy** Differential privacy has been widely used for IoT and distributed learning systems. Shokri et al., [78] presents a distributed learning system using local differential privacy without a central trusted party. However, it is only able to work on models with a small number of parameters due to its DP guarantee being per parameter. Wei et al., [79] proposed a new framework of differential privacy adding artificial noise on the client side, and discussed several key properties for differential privacy. Similar to most of the existing LDP methods [58]–[60] They add the same differential noise to all model layers, which fail to consider the range difference of model weights thus resulting in suboptimal model performance.

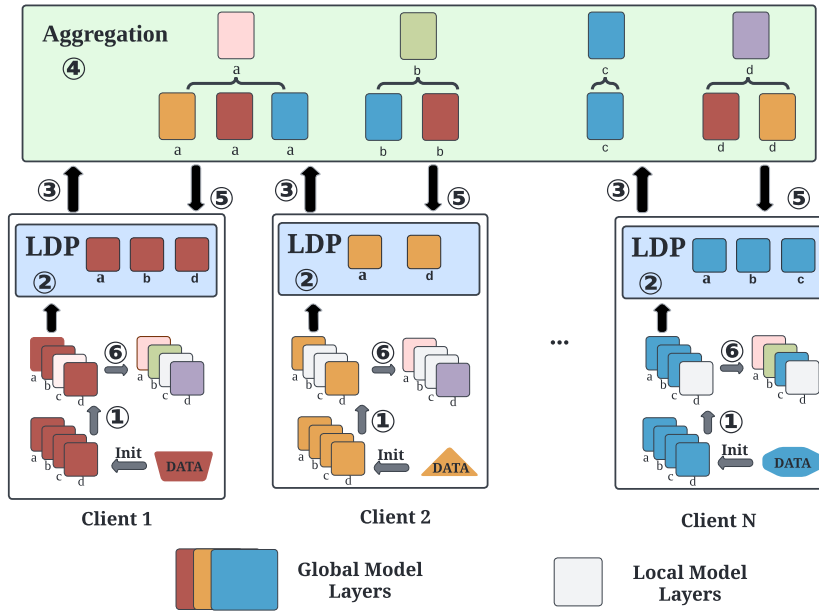
## 3.3 System Design

### 3.3.1 System Overview

In this paper, we present *Atlas*, a privacy-preserving, personalized and communication-efficient federated local differential privacy framework to achieve private and accurate IoT applications. Figure 3.3 depicts the overview of our *Atlas* framework.

In the initial phase, the central server sets up a uniform model structure for every client involved in the system. Throughout the local training iterations, we integrate a network layer importance mask to cultivate a personalized model, taking into account the significance of various layers in the deep learning model explicitly. The self-learned layer importance mask selects the layers with higher importance scores that are labeled as global layers and will be transmitted to the central server for aggregation ①. For all the selected layers, we add the weight-enhanced Gaussian Local differential privacy (GLDP) noise for each local client, taking into consideration the varying impacts of weights in each layer on the model’s performance ②. The selected layers with LDP noise together with the layers’ importance score will be transmitted to the central server for aggregation within a random time slot  $T$  to break the linkage among the model weight updates from the same clients and to mix them among updates from other clients ③. The central server then aggregates each layer based on the available layer number and sends back the updated layers to each client ④ ⑤. Each client generates their own personalized local model for IoT applications ⑥. By evaluating the impact and



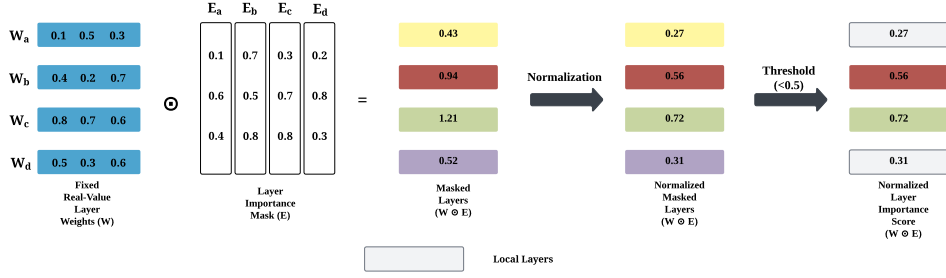


**Figure 3.3: The Overview of Atlas Framework:** ① Local IoT data training with layer importance mask for local and global layers selection; ② Dynamically adding LDP to each selected global layers by the importance of each layer. ③ Transmitting selected global layers with LDP from the devices to the central server. ④ Central server layer by layer aggregation. ⑤ Sending back updated layer information to local clients. ⑥ Personalized model generation.

significance of weights in different model layers on the model’s own performance, Atlas is able to achieve optimal model personalization. Even after incorporating local differential privacy (LDP), it still manages to deliver performance comparable to non-private methods. Additionally, by filtering the importance of model layers, we only propagate a certain number of layers in the model, which can reduce communication costs.

### 3.3.2 Model Layer Selection and Layer Importance Mask

What distinguishes Atlas from traditional approaches is that Atlas focuses on evaluating the significance of each layer in contributing to the model’s overall effectiveness, and using this evaluation to determine which layers should be transmitted to the central server. An intuitive thought on the various ranges of weights of different deep learning model layers is that *the larger the weight is, the more important the layer is*. Practically, it’s not always the truth. The relationship between the magnitude of a weight and its importance may not always be direct. In some cases, a larger



**Figure 3.4: Personalization Layer Selection via Layer Importance Mask**

weight might mean that it has a greater impact on the model’s output, which intuitively could be interpreted as the weight or the corresponding feature being more ”important.” However, the true importance of each model layer depends on multifarious factors and the actual situation could be much more complex. For example, the importance of weight also depends on its location within the network structure. In some situations [80], smaller weights might be on a critical path and have a significant impact on the output, while larger weights might be on a less important path. Therefore, it’s necessary to dynamically select the importance of each layer to perform optimal model personalization in FL. Guided by the insights of deep learning model pruning [81], we design a layer importance mask to capture the inner differences among each layer in the deep learning model. Figure 3.4 shows the detail of our model personalization layer selection via layer importance mask.

We use a fully connected layer as a representative example to demonstrate our proposed mechanism for choosing layers. It is important to mention that this approach is versatile and can be readily adapted to various other layer types. We define a fully connected layer as  $y = W \cdot x$ , where  $y \in \mathbb{R}^m$  is the output,  $x \in \mathbb{R}^n$  is the input, and  $W \in \mathbb{R}^{m \times n}$  denotes the weight matrix of the fully connected layer. Instead of using a binary mask in deep learning model pruning, we present a self-learned real-value layer importance mask. We apply the layer importance mask as the same size as the model  $W$ , and express the mask-based fully connected layer as  $y = (W \odot E) \cdot x$ , where  $\odot$  denotes the elementwise multiplication and  $E$  denotes the real-value mask, where  $E \in \mathbb{R}^{m \times n}$ . In the feed-forward step, we set  $E^B$  as a binary mask under the threshold function as Equation 3.1.

$$E_{ij}^B = \begin{cases} 1, & E_{ij} \geq \tau \\ 0, & E_{ij} < \tau \end{cases} \quad (3.1)$$

where  $E_{ij}^B$  represents the element located in  $i^{th}$  row and  $j^{th}$  column of  $E^B$ .  $\tau$  denotes the threshold. Given the non-differentiable nature of the threshold function, the current method [82] directly applies the gradients of  $E$  to the binary mask  $E^B$  as Equation 3.2.

$$\frac{\partial L}{\partial E} = \frac{\partial L}{\partial E^B} \quad (3.2)$$

While this approach can optimize  $E$  and  $E^B$ , it might lead to significant variations in gradient magnitude, potentially hindering the optimization of  $E$ . In order to mitigate the variance in the gradient, we adopt the method in [67] and integrate a sigmoid function. By replacing the hard threshold function with the differentiable sigmoid function, the back-propagation step from  $E$  to  $E^B$  can be defined as Equation 3.3.

$$\frac{\partial L}{\partial E} = \beta \odot \frac{\partial L}{\partial E^B} \quad (3.3)$$

where  $\beta$  denotes the gradient matrix of the sigmoid function.

For each training iteration, we set  $\tau$  as the threshold to determine local personalization layers and global layers. For example in Figure 3.4, as we set  $\tau = 0.5$ , we only transmit layer 2 and layer 3 to the central server for global aggregation. Since the layer importance mask is self-learned and updated with the deep learning model during the training step, we eliminate the assumption that larger weights are more important for the model. Since the pruned local contains more model information for the local user, we keep all pruned layer information for future local updates and personalized model generation, which can be beneficial for each client. On the other hand, thanks to the layer importance mask, `Atlas` only needs to transmit a partial number of model layer information to the central server, which helps to achieve better communication efficiency.

### 3.3.3 Dynamic Local Differential Privacy

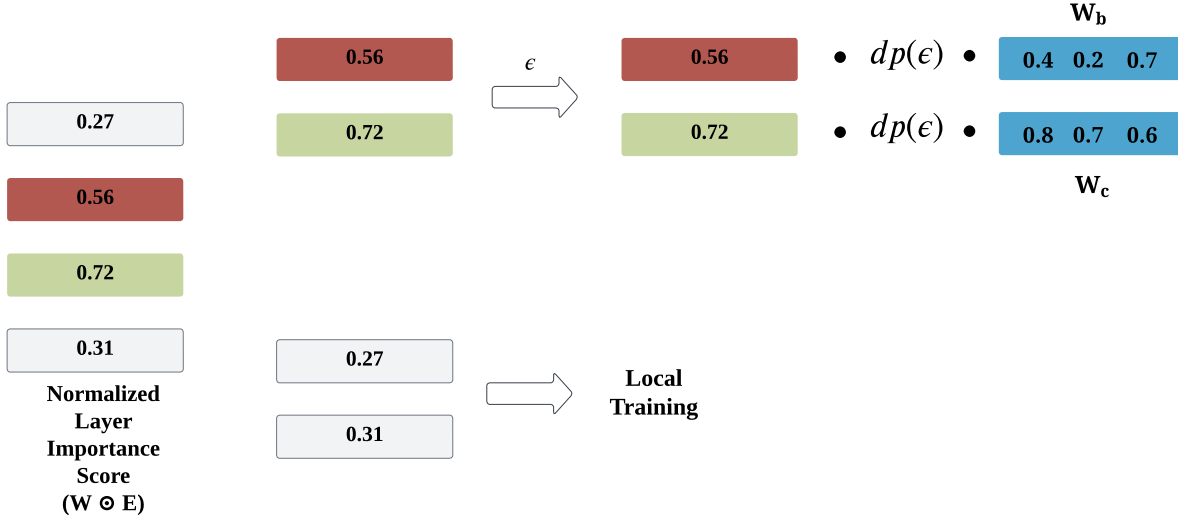
The local differential privacy module requires clipping the computed parameter gradients and adding noise (e.g., Gaussian noise). However, the depth of the layers in a deep learning model influences the features that its parameters extract. Features of different scales correspond to different layers, resulting in gradients and variances of varying magnitudes. For example, lower layers in deep convolutional networks can extract texture features, while deeper layers focus on semantic features. In most of the current local differential privacy mechanisms, there has been a lack of

explicit consideration for the variance in weight ranges across different layers of deep learning models. Recent works [54], [83] have discussed the possibilities of considering the impact of range difference of model weights in LDP, however, these methods set a fixed model weight range by default in their LDP mechanism, which is not practical. With `Atlas` pinpointing the significance of different layers in a deep learning model, we can now dynamically account for the variations in the range of model weights across these layers when implementing the local differential privacy mechanism. The core difference about `Atlas` is that `Atlas` adds different LDP noise to each selected layer based on their layer importance mask. We can also dynamically add the LDP based on the layer importance mask update from each FL global aggregation, which is optimal for the deep learning model performance while ensuring privacy guarantee. Figure 3.5 shows an example of our practical and dynamic LDP mechanism.

We first discuss the preliminary method which considers the range difference of model weights in the LDP mechanism. The main concept is to add noise separately to the gradients of each group using a unique pair of  $(C_g, z_g)$ , where  $C$  is the clipping range,  $g$  is the the model gradient. First, we divide the model weights  $w$  into  $M$  groups. In the training process, the model gradients can be represented as  $G = (g_1, g_2, \dots, g_M)$ . We denote that function of the clipping range  $C$  as  $\pi_C(\mathbf{g}) = \mathbf{g} \cdot \min(1, \frac{C}{\|\mathbf{g}\|_2})$ , for number  $L$  of training data in one training *lots*, we denote that gradient of the  $i_{th}$ , where  $1 \leq i \leq L$  training data as  $G^i = (g_1^i, g_2^i, \dots, g_M^i)$ . For all gradient  $G$ , we utilize  $(C_g, z_g)$  and perform differential privacy stochastic gradient descent (DP-SGD) as in Equation 3.4.

$$\begin{aligned} \tilde{\mathbf{G}} &= \frac{1}{L} \left[ \sum_{i=1}^L \pi_C(\mathbf{G}^i) + \mathcal{N}(0; z^2 C^2 I) \right] \\ &= \left[ \frac{1}{L} \sum_{i=1}^L \pi_C(\mathbf{G}^i) \right] + \mathcal{N}(0; \frac{z^2 C^2}{L^2} I) \end{aligned} \tag{3.4}$$

The previous research [83] proved that the above mechanism satisfied the  $(\epsilon, \delta)$ -DP, so we will not dive deep into the theoretical proof. Based on this mechanism, we observe that the differential privacy mechanism sums up the final aggregated gradients and then adds normalized noise with a variance of  $\sigma = zC$ . By averaging the obtained gradients, we acquire a gradient  $\tilde{G}$  that satisfies the requirements of differential privacy. In this case, we can reformat Equation 3.4 that for each  $g_m$ , we use independent  $(C_m, z_m)$  as the differential privacy parameter. Since we have the layer importance



**Figure 3.5: Dynamically adjust model weights to LDP mechanism.**

mask  $E$  which is So we have Equation 3.5.

$$\begin{aligned}\tilde{\mathbf{g}}_m &= \frac{E}{L} \left[ \sum_{i=1}^L \pi_{C_m}(\mathbf{g}_m^i) + \mathcal{N}(0; z_m^2 C_m^2 I) \right] \\ &= \left[ \frac{E}{L} \sum_{i=1}^L \pi_{C_m}(\mathbf{g}_m^i) \right] + \mathcal{N}(0; \frac{z_m^2 C_m^2}{L^2} I)\end{aligned}\quad (3.5)$$

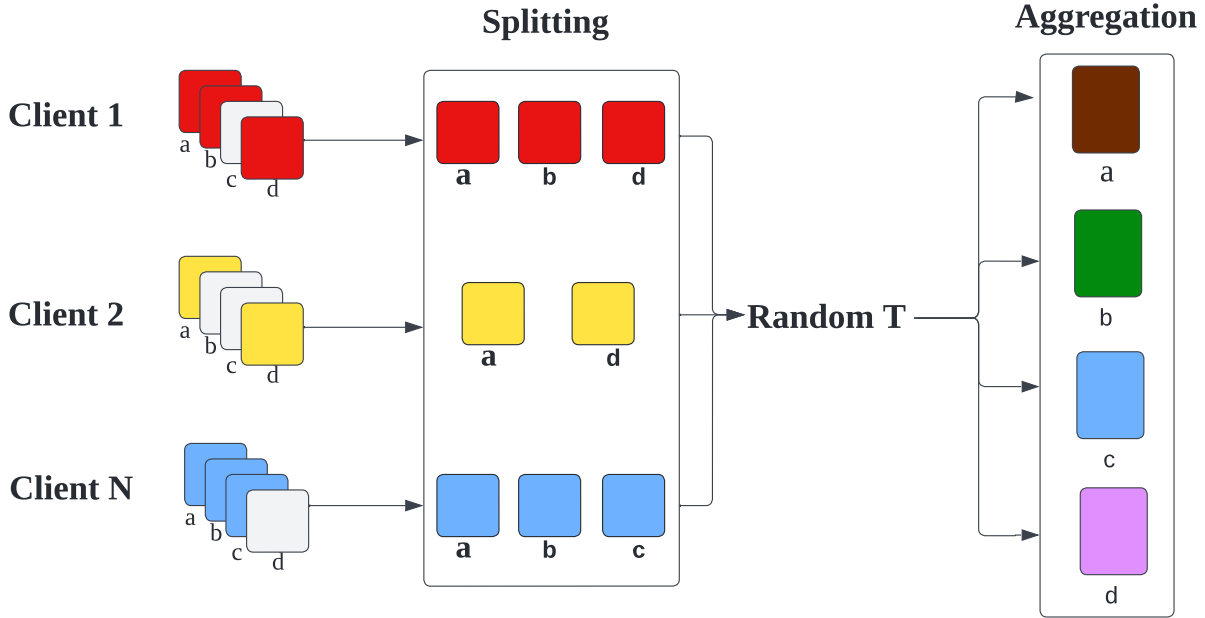
where  $E$  is the layer importance score from the previous step. Since we require  $\sum_{m=1}^M C_m^2 = C^2$ , so we have  $\|\mathbf{G}\|_2 \leq C$ , which satisfy the gradient clipping requirement of DP.

Next, we calculate the privacy bond for our dynamic LDP mechanism. We denote  $\sigma_m = E_m z_m C_m$  and reformat Equation 3.5, we get Equation 3.6.

$$\tilde{\mathbf{g}}_m = \frac{\sigma_m}{L} \left[ \sum_{i=1}^L \pi_{C_m}(\mathbf{g}_m^i) / \sigma_m + \mathcal{N}(0; I) \right] \quad (3.6)$$

We can get Equation 3.7 based on the properties of the gradient function.

$$\pi_{C_m}(\mathbf{g}_m) / \sigma_m = \pi_{C_m / \sigma_m}(\mathbf{g}_m / \sigma_m) \quad (3.7)$$



**Figure 3.6: Dynamic LDP with Parameter Shuffling**

We can further scale the input vector to  $\mathbf{G}^* = (\mathbf{g}_1/\sigma_1, \dots, \mathbf{g}_M/\sigma_m)$ , At this point, the differential privacy mechanism is equivalent to adding standard Gaussian noise to the scaled gradient vector  $\mathbf{G}^*$ , which can be expressed as Equation 3.8:

$$\tilde{\mathbf{g}}_m = \sigma_m \cdot \frac{1}{L} \left[ \sum_{i=1}^L \pi_{C_m/\sigma_m}(\mathbf{g}_m^i/\sigma_m) + \mathcal{N}(0; I) \right] \quad (3.8)$$

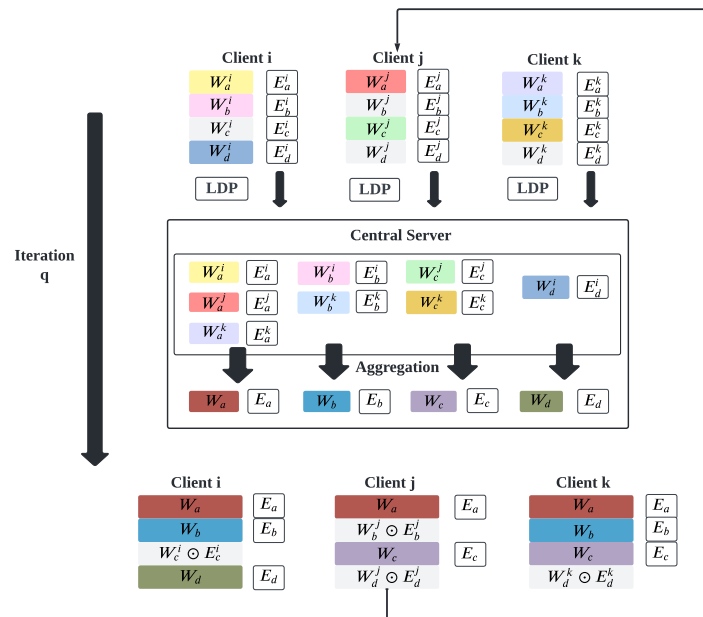
Then, we apply the clipping operation in groups with a threshold parameter of  $C_m^* = C_m/\sigma_m$ ,  $C^* = \sqrt{\sum_{m=1}^M (C_m^*)^2}$  LDP noise, where the noise multiplier  $z^*$  is as Equation 3.9.

$$z^* = \frac{1}{C^*} = \frac{1}{\sqrt{\frac{1}{z_1^2} + \dots + \frac{1}{z_M^2}}} \quad (3.9)$$

We can obtain the privacy loss associated with our dynamic LDP mechanism by substituting the noise multiplier  $z^*$  into the calculations of moments accountant with Gaussian mechanism [84].

On the other hand, prior research [85] indicates that ensuring anonymity in data reports at each

timestep and preventing their linkage over time, significantly enhances overall privacy protection. Yet, Sun et al., [54] argue that merely ensuring client anonymity doesn't adequately guard against side-channel linking attacks. For instance, if multiple clients send numerous weight updates simultaneously during each cycle, the cloud might still correlate them. In *Atlas*, even though we only transmit certain selected layers to the central server for aggregation, we understand that partial model information can also be correlated. To this end, we use LDP parameter shuffling in our system. Figure 3.6 shows an example of how we do LDP with parameter shuffling. We broadcast the selected layers within the random time  $T$  to make it more difficult for the central server to identify a single client.



**Figure 3.7: Atlas Model Aggregation**

### 3.3.4 Central Server Aggregation

Figure 3.7 shows how we perform model aggregation in the central server for *Atlas*. After a certain round of local training, each client has the locally trained model weights for each layer, together with the layer importance mask. For each global aggregation round, after adding the dynamic LDP mechanism to the selected model weights and layer importance mask, the central

server performs model aggregation. Note that for each client, the number of the selected layers is different, which means we can not directly perform an averaging strategy like FedAvg. To solve this problem, we perform layer-by-layer aggregation. As Figure 3.7 shows, client  $i$  selects model weight  $W_a^i$ ,  $W_b^i$  and  $W_d^i$  for global aggregation, and keeps  $W_c^i$  locally for personalization. Client  $j$  selects model weight  $W_a^j$  and  $W_c^j$  for global aggregation and keeps  $W_b^j$  and  $W_d^j$  locally for personalization. Client  $k$  selects model weight  $W_a^k$ ,  $W_b^k$  and  $W_c^k$  for global aggregation, and keeps  $W_d^k$  locally for personalization. Since each client has the default model structure, for each layer and client, we only aggregate their transmitted layers for global aggregation. More specifically in this example, we have:

$$\begin{aligned}
 W_{a,t+1} &= \frac{W_{a,t}^i + W_{a,t}^j + W_{a,t}^k}{3} \\
 W_{b,t+1} &= \frac{W_{b,t}^i + W_{b,t}^k}{2} \\
 W_{c,t+1} &= \frac{W_{c,t}^j + W_{c,t}^k}{2} \\
 W_{d,t+1} &= W_{d,t}^i
 \end{aligned} \tag{3.10}$$

where  $W_t$  is the model weights for different layers before global aggregation,  $W_{t+1}$  is the model weights for different layers after global aggregation. We perform the same strategy to the layer importance mask as well.

### 3.3.5 Personalized Model Generation

As shown in Figure 3.3, after each global aggregation round, the central server sends back the updated layer weight for each client. Before the next local training round, each client generates their personalized modal locally based on the locally pruned layers and globally updated layers. Figure 3.12 shows an example of how `Atlas` achieves personalized model generation. We take client  $i$  as an example. Since the layer importance mask determined  $W_{a,t}^i$ ,  $W_{b,t}^i$  and  $W_{d,t}^i$  for global aggregation,  $W_{c,t}^i$  stayed locally waiting for the next local computational round. After our system updates all the selected layers, Client  $i$  receives the new weights for different layers, which is  $W_{a,t+1}^i$ ,  $W_{b,t+1}^i$  and  $W_{d,t+1}^i$ . The updated model weights then combine with the localized  $W_{c,t}^i$  for the next local computational round for local training. Such a design takes into account the global aggregation as well as the localized personalized information, which ensures that the updated personalized model can achieve the best performance for each client in the system.



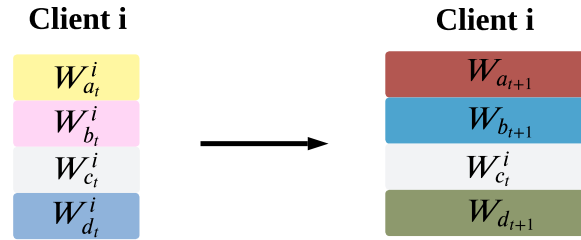


Figure 3.8: Atlas Personalized Model Generation

## 3.4 Evaluation

To demonstrate the generality, practicability and usefulness of Atlas across different IoT applications, we design our experiments to evaluate Atlas on five real-world IoT application datasets focusing on the following research questions:

- **RQ1:** How does Atlas perform on real-world datasets compared to traditional methods, non-privacy-preserving FL methods, and LDP-based FL methods?
- **RQ2:** How does Atlas perform on variate of scalability compared to non-privacy-preserving FL methods and LDP-based FL methods?
- **RQ3:** How do different LDP privacy budgets in Atlas and LDP-based FL methods affect the accuracy of different IoT applications?
- **RQ4:** For IoT applications, it's practical to have new clients with heterogeneous data joining the system. How well can Atlas handle the new clients?
- **RQ5:** How is the communication efficiency?

### 3.4.1 Applications and Datasets

Below are detailed descriptions of these five IoT applications, the statistics of the datasets that are used in the applications are concluded in table 3.2.

**Application#1: Energy Prediction.** Pecan Street <sup>1</sup> contains the energy consumption for every second of different home appliances in 1641 residents in Texas from 2013-01-01 to 2017-12-31. Since the recorded data has some missing data points, also the devices in each resident are not exactly the same, we randomly select 100 residents that have the full load records as our default experiment dataset. Recently, Pecan Street dataset also updated second-level data for New York and minute-level data for California. To test how `Atlas` can handle the new users and demonstrate the performance of **RQ4**, we will use the New York dataset together with Texas dataset for evaluation.

**Application#2: Human Activity Recognition.** Human Activity Recognition (HAR) is a task that distinguishes a physical activity completed by an individual subject. Physical activities, for example, *Walking*, *Standing*, and *Sitting*, etc. build our comprehensive actions in daily life. A well-known publicly available HAR dataset called *Human Activity Recognition Using Smartphones DataSet* <sup>2</sup>, the HAR using smartphones dataset contains 30 participants age ranged from 19-48. Each person performed six activities (WALKING, WALKING UPSTAIRS, WALKING DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist [46]. Two types of 3-axial signal data are collected from the accelerometer sensor and gyroscope sensor in fixed-width sliding windows of 2.56 sec respectively.

**Application#3: Sleep Stage Classification.** Sleep stage classification is essential for the assessment of sleep quality and the diagnosis of sleep disorders. The *Montreal Archive of Sleep Studies (MASS)* <sup>3</sup> is an open-access and collaborative database of laboratory-based polysomnography (PSG) recordings. There are 97 males and 103 females of age ranging from 18 to 76 years participated in this study, and five sleep stages are recorded for classification [86].

**Application#4: Daily Activity Recognition.** The WISDM (Wireless Sensor Data Mining) dataset [87] stands as a prominent resource for daily activity recognition tasks. It utilizes accelerometer and gyroscope sensor data gathered from smartphones and smartwatches. The dataset encompasses data from 51 participants, each engaged in 18 distinct daily activities over sessions lasting 3 minutes. We keep the same data pre-processing mechanism as mentioned in [88]. More specifically, we have streamlined the dataset by consolidating activities like eating soup, chips, pasta, and sandwiches into a unified "eating" category, while eliminating rare activities associated with ball play, such as kicking, catching, and dribbling. Acknowledging that individuals might not simultaneously

---

<sup>1</sup><https://www.pecanstreet.org/dataport/>

<sup>2</sup><https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones>

<sup>3</sup><http://ceams-carsm.ca/mass/>

carry a smartphone and wear a smartwatch in real-world scenarios, we divided WISDM into two distinct datasets: WISDM-W, containing solely smartwatch data, and WISDM-P, comprising only smartphone data. The training and test sets for WISDM-W include 16,569 and 4,103 samples, respectively, while for WISDM-P, they contain 13,714 and 4,073 samples, respectively. In our experiment, we only focused on the WISDM-W subset of this dataset, mainly because the application#2 is a smartphone-based dataset.

**Application#5: Smart Home Daily Living Sequence Recognition.** The CASAS dataset<sup>4</sup>, a product of the CASAS smart home project, serves as a resource for recognizing daily living activities (ADL) through sequences of sensor states over time, aiming to facilitate independent living applications. The data collection took place across three different apartments, each outfitted with a trio of sensor types: motion, temperature, and door sensors. We keep the same data pre-processing mechanism as mentioned in [88]. We have chosen five specific datasets named “Milan”, “Cairo”, “Kyoto2”, “Kyoto3”, and “Kyoto4”, selected for the consistency in their sensor data representation. Within each dataset, we have streamlined the original ADL categories into 11 home activity-related categories such as “sleep”, “eat”, and “bath”. Each data entry is a categorical time series with a length of 2,000, depicting sensor states across a certain time span. In total, the training and test sets comprise 12,190 and 3,048 samples, respectively.

**Application#6: Image Classification.** Image classification is an important part of smart IoT applications. We use CIFAR-10 [89] to demonstrate the performance of *Atlas* in Image Classification. Although CIFAR-10 is not a typical IoT application dataset, it has been widely used to test the performance of federated learning frameworks. In this experiment, we use Resnet 32 [74] as the base model for training.

**Table 3.2: Dataset Statistics.**

Dataset	IoT Platform	Statistics		
		Data Modality	Dataset Size	Model
Pecan Street	Smart Home	Energy	3.2 GB	LSTM
UCI-HAR	Smartphone	Accelerometer, Gyroscope	58.17 MB	LSTM
MASS	PSG	PSG	1.2 GB	LSTM
WISDM-W	Smartwatch	Accelerometer, Gyroscope	294 MB	LSTM
CASAS	Smart Home	Motion Sensor, Door Sensor, Thermostat	233 MB	LSTM
CIFAR-10	-	Image	163 MB	Resnet 32

<sup>4</sup><https://casas.wsu.edu/datasets/>

### 3.4.2 Experimental Setup and Evaluation Metrics

We implemented our experiments on five real-world IoT application datasets using two NVIDIA RTX 3090 GPUs. We divided the data into training and test sets using an 80-20 split. We chose the base model with the same structure including 8 LSTM layers with three fully connected layers for each application (for CIFAR-10, we use Resnet 32 as the base model). By default, each participating client performs 10 local training rounds for one global aggregation round when evaluating the model performance of each application. We also evaluate the performance of different participating clients and the influence on different local training rounds/global aggregation rounds as well.

We compare `Atlas` with 8 baselines in 3 categories. (1) Traditional methods include localized, centralized model, centralized clustering and FedAvg, (2) Personalized FL approaches include FedPer and Cluster FL), (3) Privacy-preserving FL methods include LATENT, PM-SUB and HMTTP.

- **Localized model**, trains models on local clients by using only the local data, without the collaborations between different clients. Since there is no privacy issue nor communication cost when training data is completely local, we use this method only as a baseline to compare the model performance among different IoT applications.
- **Centralized model**, trains all data on a centralized global server as a unified model. This method has significant privacy issues such as malicious cloud servers, etc. We did not use any privacy protection mechanisms in this approach. We use this method only as a baseline to compare the model performance among different IoT applications.
- **Centralized clustering model [90]**, trains all data on a centralized global model with a centralized clustering algorithm. Similar to the Localized and centralized model, we use the centralized clustering model only as a baseline to compare the model performance among different IoT applications. We include this method to show the best performance that collaborative training among different users can get to compare with our method.
- **FedAvg [47]**, is a classic federated learning model that clients exchange model information with the central server by sending updated local parameters and retrieving the aggregated global model, facilitating ongoing local training.

- **FedPer** [29], is a federated learning model that each client contributes to share their lower layers and leave their upper layers locally for personalization. The method requires to pre-set the number of layers that are shared in the system. In our experiments, we set the number of layers to be 4.
- **Cluster FL** [66], introduces a unique clustered federated learning framework, enhancing the training accuracy while discerning the inherent clustering relationship among data across different nodes. Leveraging these cluster relationships, ClusterFL efficiently eliminates slower-converging or weakly-correlated nodes within each cluster, hastening convergence without sacrificing accuracy.
- **FedSGD-LDP** [71], introduces Gaussian mechanisms to preserve local differential privacy (LDP) of user data in the FL model. The method directly adds LDP noise to all the model weights to achieve privacy guarantee for each client.
- **LDP-FL** [54], makes the local weights update differentially private by adapting to the varying ranges at different layers of a deep neural network, which introduces a smaller variance of the estimated model weights. This method considers a variety of range differences in model weights to achieve a better performance.

For evaluation metrics, we use accuracy for the classification IoT applications and  $R^2$  for regression applications. We evaluate the performance on each client’s test data, and report the average accuracy over all clients for evaluations. For communication evaluation, we normalize the time cost as the ratio to the communication time of FedAVG as reported communication cost.

### 3.4.3 Results

#### 3.4.3 RQ1: Performance Analysis on real-world IoT data

In this section, we evaluate how `Atlas` performs compared to the other 8 baseline models. Figure 3.9 demonstrates that `Atlas` can achieve comparable accuracy performance compared to the other personalized model while providing privacy guarantee. Also, `Atlas` outperforms all other LDP-based FL frameworks. More specifically, we can observe that for UCI-HAR, `Atlas` can achieve 94% accuracy, which is similar level compared to central training, and FedPer. Compared to the best performance model, which is ClusterFL, `Atlas` sacrifices 4% accuracy performance

in exchange for privacy-preserving, which is practical for real-world IoT applications. For MASS, `Atlas` achieves 85% accuracy which sacrifices 4% accuracy compared to the centralized clustering method and 3% compared to `ClusterFL`. `Atlas` also outperforms the LDP-FL-based model by 7%. For WISDM-W and CASAS, the results follow the same pattern as the classification tasks. We observe that `Atlas` generally sacrifices 2% to 6% of accuracy in exchange for privacy guarantee. One unique case is the Pecan Street data, we observe that `Atlas` sacrifices 0.104 in  $R^2$ , which is larger than other tasks. We also noticed that for all LDP-based FL methods, the performance dropped dramatically. The reason is that time-series data such as energy data, typically contains crucial temporal correlations and patterns. Adding noise to satisfy differential privacy requirements can distort these temporal relationships, leading to misinformation extracted from the data which makes the model performance drop higher. For all LDP base FL frameworks, `Atlas` outperforms LDP-FL by 0.142 and FedSGD-LDP by 0.272, which is still the best for privacy-guarantee methods. For image classification tasks like CIFAR-10, we observe the same trend as the previous tasks. We also notice that the accuracy performance of `Atlas` only drops 3.2% comparing to the best performed method, which is central cluster, and outperformed the LDP-based methods by 10%.

### 3.4.3 RQ2: Performance analysis on scalability.

Figure 3.10 shows the model performance comparison on different numbers of clients participating in the system. We compare `Atlas` with non-privacy-preserving FL methods, LDP-based FL methods in this section. For each IoT application, we assign a different number of clients during the training process. We observe that the general performance of `Atlas` follows the same pattern as in RQ1. Interestingly to see that, for UCI-HAR, the performance changes dramatically based on the change in the number of clients compared to other datasets. We see that along with the growth of the participating clients, most of the methods first achieve a higher model performance, but fail to hold such a performance while the number of clients is growing higher. However, `Atlas` and `ClusterFL` remain steady with the growth of the number of clients. The reason is because of the model personalization. For other methods, since there is no model personalization mechanism, as more and more clients joined the system, the diversity of clients' data makes the global model vulnerable to providing an accurate global model for each client. However, `Atlas` utilizes the layer personalization mask for model personalization, which keeps the personalization layers locally and only updates the weights that are beneficial for the global model, which makes `Atlas` maintain

a steady performance facing a larger amount of clients. From this study, we also think that it showcases the uniqueness of the IoT application. For different IoT applications, we can observe different moving patterns with different numbers of users. As for the other applications, we can see that for MASS, Pecan Street, CASAS, WISDM-W and CIFAR-10, the accuracy and R square first move higher along with the increase of the client numbers by around 10%, then decrease by around 2-4% once the number of clients is getting larger, which is around 70. The main reason is that, the clients may have different data patterns, and since the number of clients is going up, the performance will slightly drop due to the heterogeneity of user data.

### 3.4.3 RQ3: Impact of Different LDP Parameter $\epsilon$

Figure 3.11 shows the impact of privacy budget  $\epsilon$  for LDP-based FL framework. In differential privacy, there's typically a trade-off between privacy and accuracy. A larger privacy budget allows for more accurate results, as it permits more noise to be added to the data, thereby preserving privacy while minimizing the impact on the accuracy of the analysis or query results. We assign the same  $\epsilon$  to all LDP-based FL models to ensure that they are under the same privacy guarantee level. We can observe that for all IoT applications, the accuracy follows Noise-Free  $\epsilon_{Atlas}$  > LDP-FL > FedSGD-LDP. We can also observe that the accuracy remains steady once the privacy budget increases to 2. *Atlas* outperforms all other baseline models and only sacrifices 2% to 6% compared to the noise-free method.

### 3.4.3 RQ4: Impact on New Users

In real-world settings, it's practical that there will be new users joining the system. Traditionally researchers tend to use dataset separation to evaluate the model's ability to handle new clients in the system. However, it's not practical in real-world IoT settings. Because even with dataset separation, part of the separated dataset still contains the data pattern even if it's not used in either model training or testing. To this end, we want to evaluate how *Atlas* handles new users more practically. As we mentioned in the dataset selection section. We trained our model solely using the Pecan Street energy datasets from houses in Texas. To evaluate our assumption, here we use the trained model from Pecan Street Texas data and use purely Pecan Street New York data as testing. Since there is a time difference between NY and TX, we adjusted the timeframe in the original dataset and performed the experiment. Figure 3.12 shows the result of the impact on new users joining the

system. We can observe that, compared with the local training method, `Atlas` can achieve better performance for most of the new clients, which proves that `Atlas` can handle new users joining the system.

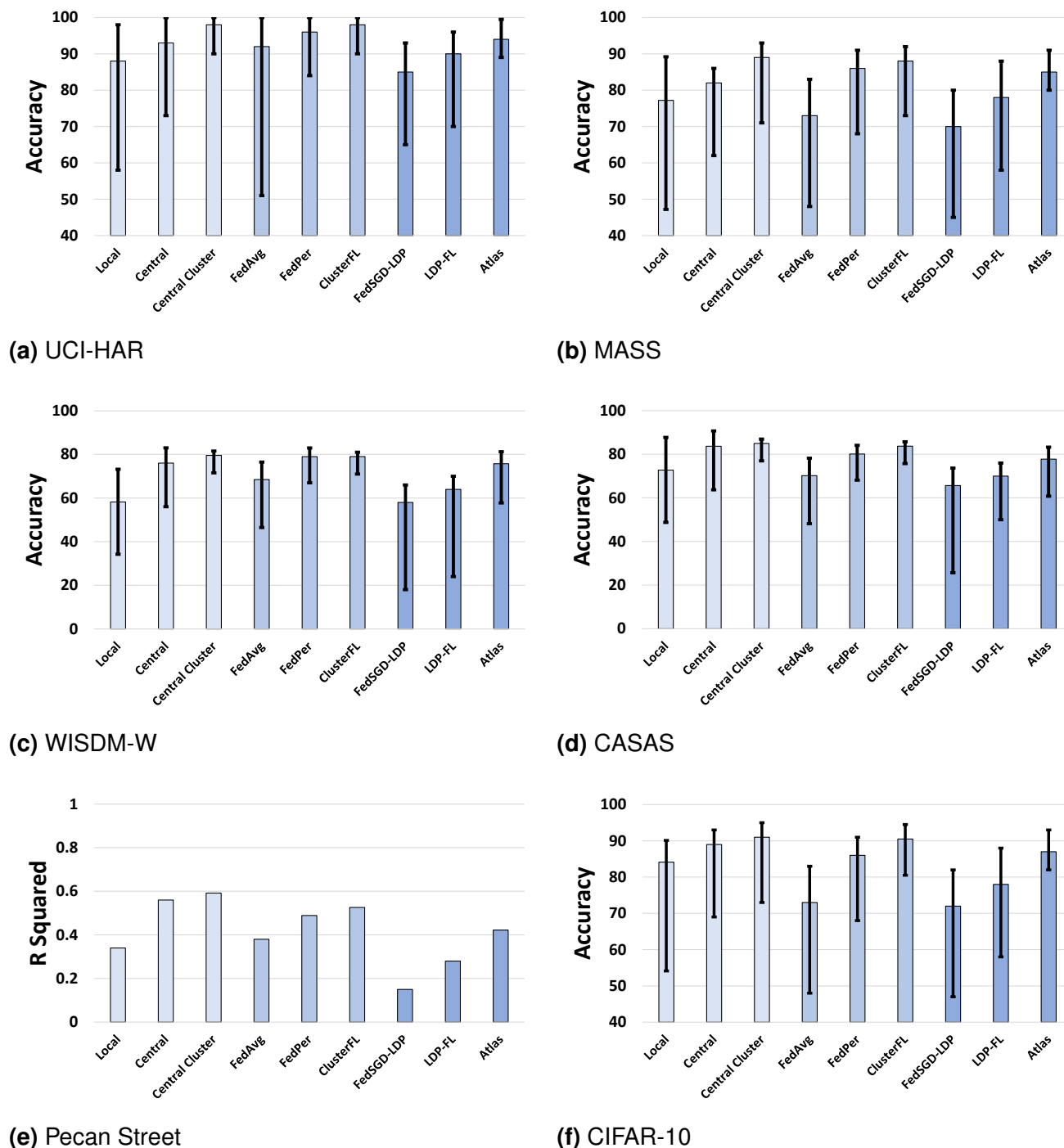
### 3.4.3 RQ5: Communication Efficiency

Figure 3.13 shows the performance of communication efficiency between FL methods and `Atlas`. We observe that for all IoT applications, the results have a similar pattern that `Atlas` outperforms FedAVG and ClusterFL, but not as well as FedPer. The reason is that, although both FedPer and `Atlas` pruned certain model layers locally and only transmit partial of the whole model, `Atlas` also needs to transmit the layer importance mask to dynamically the layer selection mechanism and LDP, which will incur more communication cost.

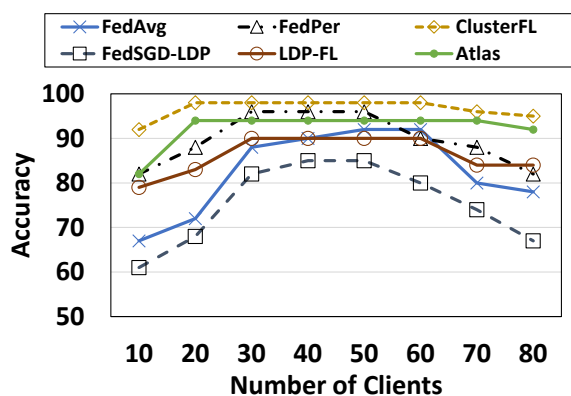
## 3.5 Conclusion

We present `Atlas`, a practical personalized federated learning framework with dynamic local differential privacy for IoT applications. Initially, we implemented a layer importance mask for sharing layers, distinguishing between global and personalized components within the local model. Following that, we introduce weight-enhanced Local Differential Privacy (LDP) noise to the global layers prior to their integration into the federated learning framework for aggregation purposes. Ultimately, we amalgamate the local personalized layers with the aggregated global layers to effectively execute IoT tasks. `Atlas` achieved comparable model performance with non-privacy-preserving methods with privacy guarantee.

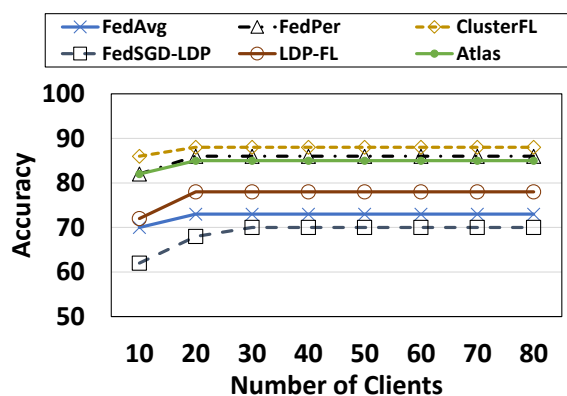




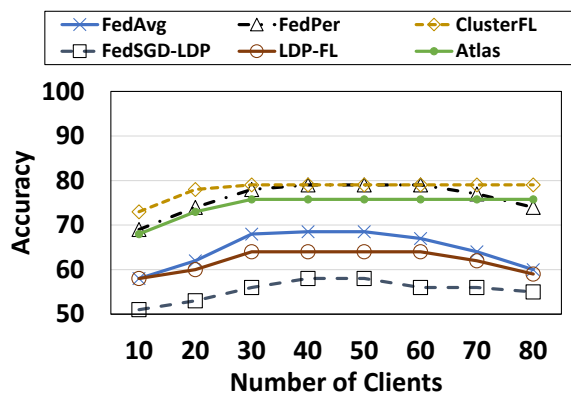
**Figure 3.9: Performance comparison on accuracy and  $R^2$  between traditional methods, non-privacy-preserving FL methods, LDP-based FL methods and Atlas**



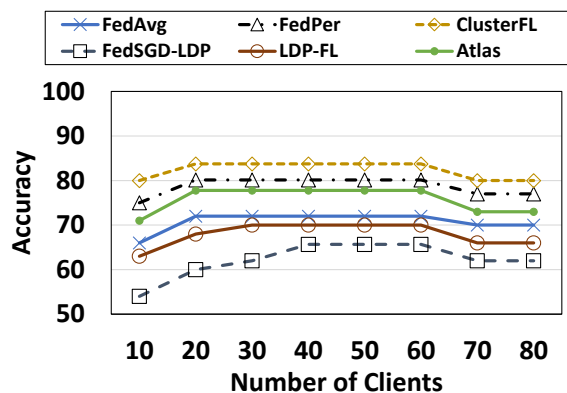
(a) UCI-HAR



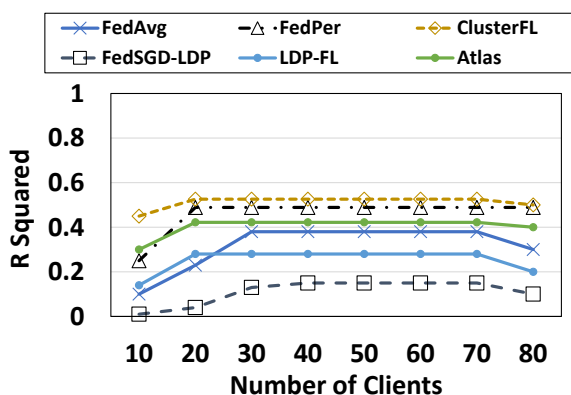
(b) MASS



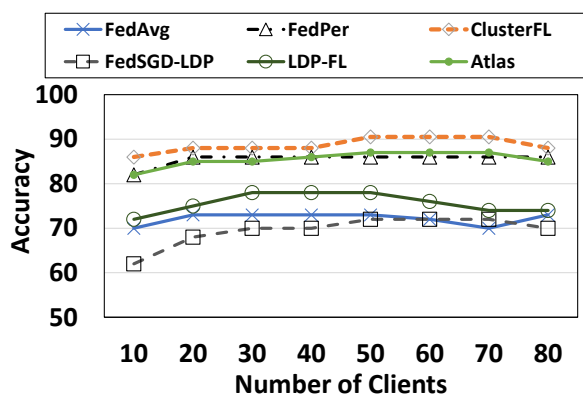
(c) WISDM-W



(d) CASAS

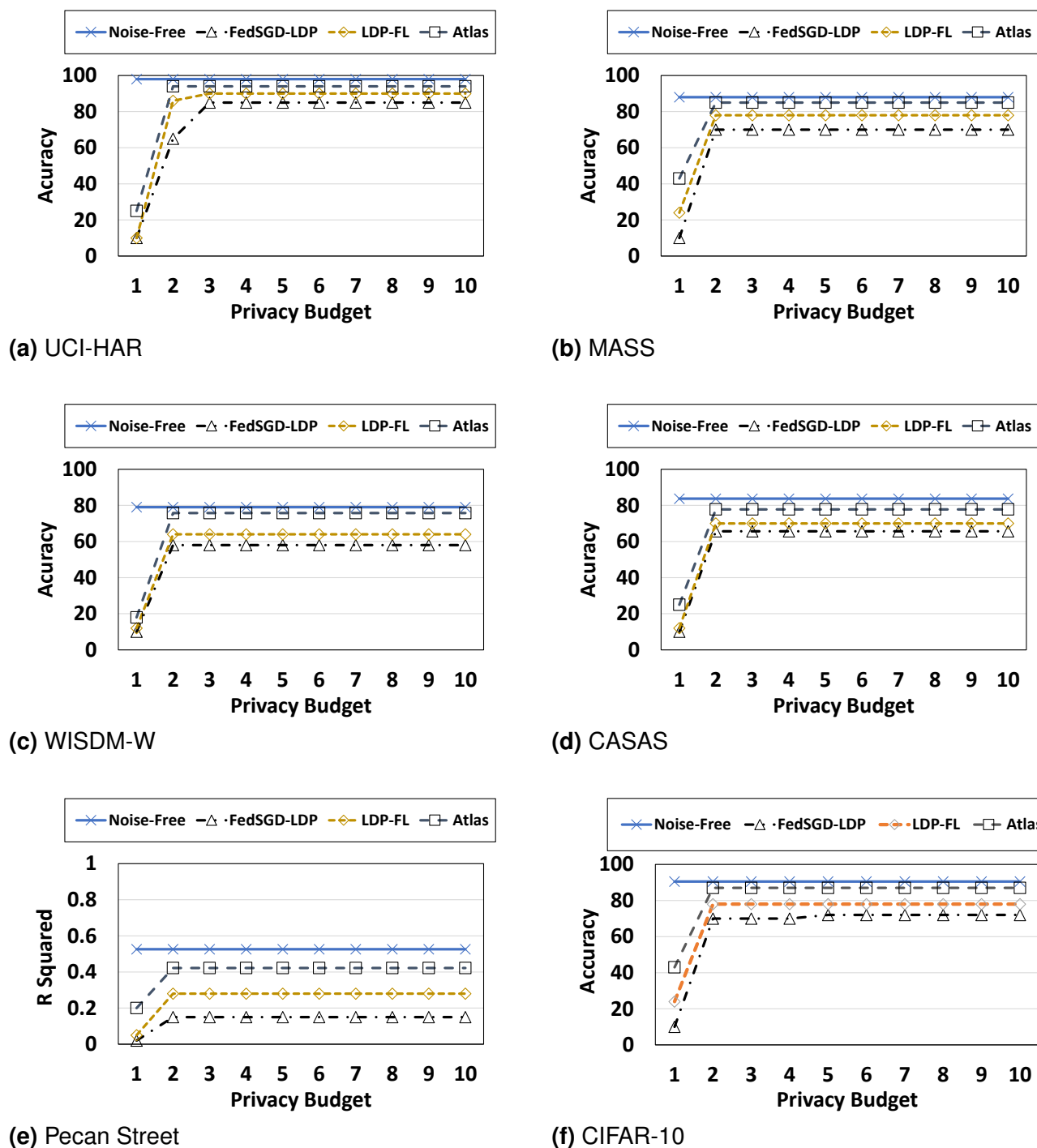


(e) Pecan Street

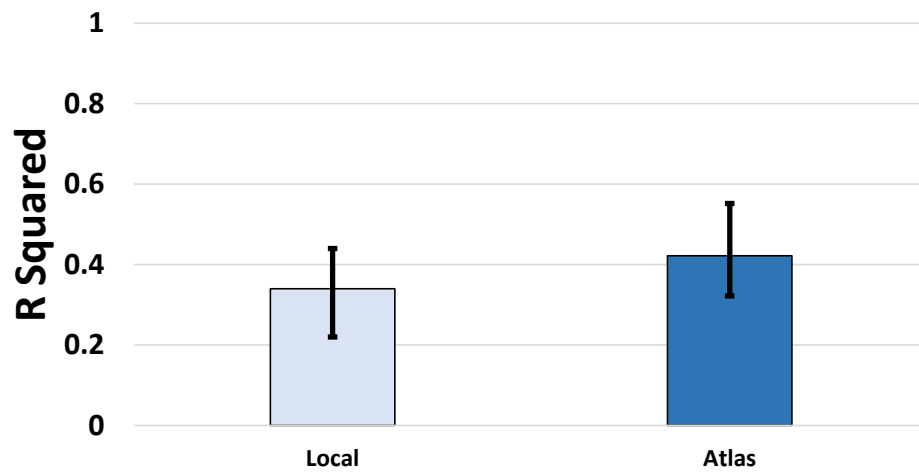


(f) CIFAR-10

**Figure 3.10: Model performance comparison on variate of scalability between non-privacy-preserving FL methods, LDP-based FL methods and Atlas.**



**Figure 3.11: Model performance comparison on variate of privacy budget  $\epsilon$  between LDP-based FL methods and Atlas.**



**Figure 3.12: Model performance comparison on adding new users in Atlas.**

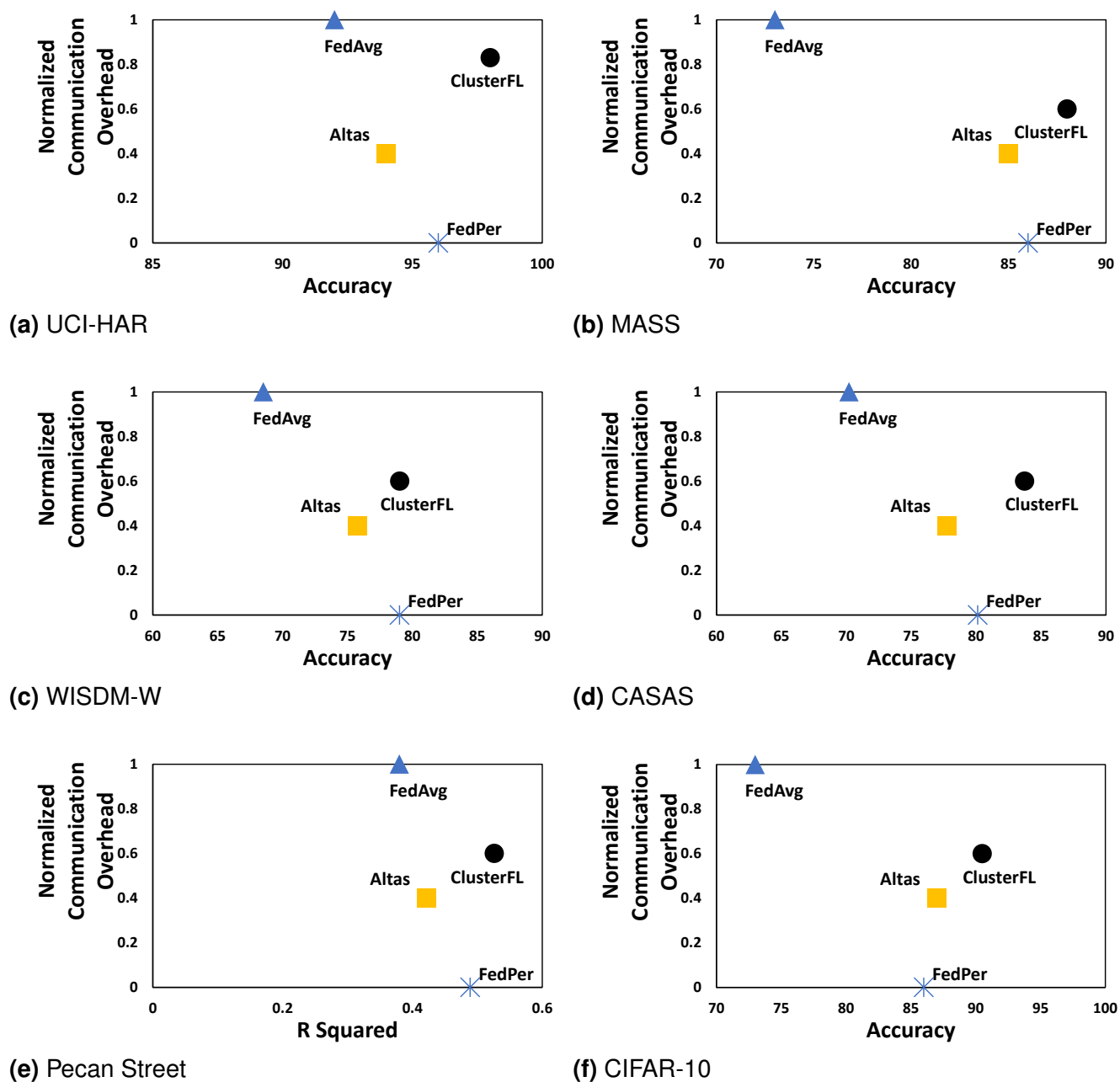


Figure 3.13: Model performance comparison on communication efficiency between FL methods and Atlas.

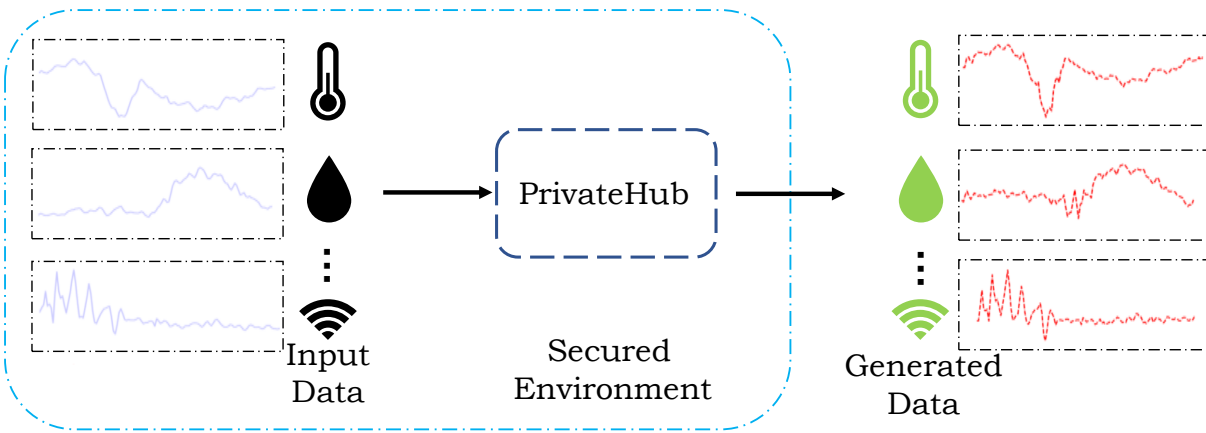
## Chapter 4

### **PRIVATEHUB: CONTRASTIVE DIFFUSION MODEL FOR PRIVATE MULTI-SENSOR SCENARIO DATA GENERATION**

Internet of Things (IoT) systems have become integral to today's digital society. These systems, by connecting various sensors, enable intelligent applications such as traffic forecasting [44], health monitoring [45], and human activity recognition (HAR) [46]. Researchers have done good work utilizing the rich amount of IoT sensors to empower these applications, however, it also brings sensor data privacy challenges.

When it comes to sensor data privacy, we often tend to contemplate the issue solely from the perspective of the data stream itself. For instance, when the data stream originates from users' bodily information, it is commonly perceived as privacy-sensitive data [91]. However, in a complex environment such as multi-sensor scenarios, data amalgamated from multiple sensors can also reveal a plethora of privacy-sensitive information, even if each sensor's data stream individually does not appear to be privacy-sensitive. For example, in a smart space environment surrounded by multiple sensors [92], such as smart thermostats and inexpensive commercial off-the-shelf sensors like Doppler, temperature, and humidity sensors, these sensor streams can actively infer various types of indoor activities. Although some activities may be considered as non-private activities (e.g., standing, sitting), other activities may be considered as highly private activities (e.g., calling via Zoom, typing on the computer, eating meals, writing on the whiteboard). Ensuring that non-private activities are accurately identified while private activities remain concealed has emerged as a new privacy challenge in the context of multi-sensor IoT environments. In addition, with the increasing number of sensors, such privacy concerns are becoming increasingly pronounced.

There are three kinds of traditional privacy protection methods for IoT data streams. (1) **Rule-based privacy frameworks (data access restriction policies)** [93]–[95] establish a set of rules or



**Figure 4.1: High-level architecture of PrivateHub**

policies that govern the handling of personal data. These rules can include requirements such as obtaining explicit consent from individuals before collecting their data, limiting access to sensitive information to authorized personnel only, encrypting data during transmission and storage, and specifying conditions under which data can be shared with third parties. However, rule-based privacy frameworks depend on human formulation and enforcement of rules, making them susceptible to human errors, misunderstandings, or negligence. Unclear or inaccurate rules, or improper enforcement, can lead to improper data processing or non-compliance with regulatory requirements. On the other hand, the rule-based privacy framework is not suitable for our new privacy challenge, as it processes data as per data stream instead of the interaction of multiple data streams, which is the main reason that causes privacy issues. Also, if the users send multiple data streams to a third party, the third party can still run activity inference models to detect private activities. (2) **Differential privacy based frameworks** [68], [96], [97] employ differential privacy (DP) methods to provide privacy protection for sensor data as well. Differential privacy methods add noise (e.g., Laplace Noise, Gaussian Noise) to the data in a mathematically rigorous manner with a privacy budget  $\epsilon$  [55] to provide strong guarantees of privacy for a data stream. Such a unified method is suitable for a single-sensor data stream, however, not ideal for complex multi-sensor scenarios. The reason is that, in complex multi-sensor scenarios, DP achieves privacy protection by adding noise to all data streams based on  $\epsilon$ , which fails to capture the inner connection among sensors in a multi-sensor environment, resulting in a degradation of all applications no matter they are considered as private or non-private. (3) **Generative models for data privacy** [98]–[100] also serve as a way to protect the

privacy of IoT data by enabling the generation of synthetic data similar to the original datastreams without exposing the raw data. However, traditional methods such as VAE [101], GAN [102], and conditional GAN [103] suffer from poor performance caused by posterior and mode collapse, where the model ignores certain factors in the latent variable, and the generator focuses on generating only a few modes of the data distribution, neglecting others. Compared to the aforementioned methods, diffusion models [104] exhibit greater stability and are capable of producing high-quality samples characterized by enhanced clarity and realism. This is attributed to their capacity to replicate the data generation process continuously over time. However, in addressing this novel privacy challenge, it is imperative to generate data streams conditionally to accurately discern non-private applications while safeguarding the privacy of private applications, which is presently beyond the capability of the current diffusion models.

Given this new privacy challenge caused by multi-sensor IoT environments, the ideal solution should be able to achieve the following three requirements: (1) Accurate identification of non-private applications; (2) Successfully concealing private applications; (3) the sensor datastreams are meaningful. (e.g., smart thermostats data can still perform temperature prediction.)

In this paper, we propose `PrivateHub`, a novel synthetic datastreams generation method that utilizes contrastive learning in the diffusion model to conditional generating datastreams within the realm of private multi-sensor scenarios, ensuring accurate identification of non-private applications while simultaneously maintaining the concealment of private applications. `PrivateHub` is driven by the objective of harnessing the potent generative capacity of diffusion models to discern the aforementioned data types. Figure 4.1 shows the high-level architecture of `PrivateHub`. Our goal is that: for all the multi-sensor data streams, the users can use them as input and send them into our proposed framework in a secure environment. Then, the output, which is the generated synthetic data streams, can be used to detect all non-private sensitive applications and fail to detect any private applications labeled by the data owner.

Specifically, we propose a comprehensive framework for enhancing the performance of diffusion models in generating task-specific data, comprising two key stages: App-Conditioned Pre-training (ACP) and APP-Aware Fine-tuning (AAF). Initially, ACP involves pre-training a diffusion model using standard procedures on multi-sensor datastreams with application category embedding, enabling the model to generate app-conditioned data. Subsequently, AAF facilitates the diffusion model in further discriminating between privacy-sensitive and non-privacy-sensitive data through



contrastive learning optimization. In this stage, an off-the-shelf application classifier is employed to extract features from various generated data, encouraging the model to produce data exhibiting feature distances closer to those of privacy non-sensitive ones. We conduct our experiments on real-world multi-sensor datasets from both smart homes and smart offices. We first run a motivation task to prove that this multi-sensor environment application privacy is severe and important, then we showcase that our proposed method can successfully solve this emerging problem and also outperformed the other comparison methods.

## **4.1 Related Work**

### **4.1.1 Data Privacy**

Numerous studies have explored data privacy from different perspectives. There are mainly two different types: Rule-based Privacy Frameworks [93]–[95] and Differential privacy based frameworks [68], [96], [97]. For rule-based frameworks, Tamini et al., [105] created a user-centric privacy protection strategy focusing on user habit-based anomaly detection system. Lola et al., [93] designed a two phase principles. First, the IoT device manufacturer declares their device’s data collection intentions. Second, the user declares their own preferences of what is permitted to the IoT device. Nonetheless, these frameworks rely on the human creation and enforcement of rules, leaving them vulnerable to human errors, misunderstandings, or oversight. Ambiguous or incorrect rules, or their improper application, can result in inappropriate data handling or failures to meet legal standards. Furthermore, rule-based frameworks are ill-suited to our new privacy challenge, as they manage data by individual streams rather than considering the interactions between multiple streams, which are central to the privacy issues in question. For DP based systems, Gao et al., [68] tried to integrate federated learning and LDP to ensure IoT data privacy. However, while effective for single-sensor streams, this approach is less effective in complex multi-sensor scenarios. DP-based approaches do not account for the intricate relationships among sensors in a multi-sensor environment, resulting in diminished utility across all applications, regardless of their privacy designation.

### 4.1.2 Generative Models

Generative models are often employed for data augmentation tasks. Based on the generated data, generative models have huge potential to contribute to overcoming the multi-sensor data privacy challenge. GAN-based methods are increasingly effective in generating realistic time-series data. For example, SensoryGANs [106] has introduced systems for creating synthetic sensor data, which significantly enhance human activity recognition in resource-constrained settings. TimeGAN [107] and ActivityGAN [108] excel at preserving temporal dynamics and enriching sensor-based HAR datasets, respectively. However, GAN-based methods experience diminished performance due to posterior and mode collapse, where the model overlooks specific aspects of the latent variables and the generator predominantly creates only a limited range of the data distribution's modes, disregarding the rest.

Diffusion models are suitable for generating synthetic IoT data streams with good quality. For example, Sivaroopan et al., [109] presented a Diffusion-Model (DM) based end-to-end framework, NetDiffus, for synthetic network traffic generation. However, most of the conditional generation in diffusion models [110], [111] mainly focuses on image generation, which is not suitable for multi-sensor data streams.

Among all smart building applications, the data generated by IoT sensors contains unique user-specific patterns that can reveal the user's identity. Thus, data utility can enable important applications, but may also lead to user's privacy concerns. As demonstrated in earlier work [112], speech data gathered by the built-in microphones of virtual personal assistants and mobile phones may be processed to infer the user's emotional state and mental health status without the user's awareness or consent. Cameras integrated in mobile devices, particularly in indoor situations, can gather sensitive and personal data [113]. On the other hand, if the application makes an uninvited or invasive inference on this data, or when this data is transferred to third-party servers for storage and processing, this might result in privacy breaches, such as building occupancy detection using smart meter data [114].

Various research works have already been proposed to solve data privacy issue in several aspects, such as differential privacy algorithms [115], GANs, Autoencoders [116], and Variational Autoencoders(VAE) [101]. Hajihassnain et al. [117] introduced ObscureNet, an encoder-decoder architecture that successfully masks private characteristics connected with time series data collected by sensors in IoT devices while retaining the original time series' information content. Osia et

al. [118] presented a new hybrid framework for efficient privacy-preserving mobile analytics by breaking down a DNN into a feature extractor module, which should be deployed on the user's device, and a classifier module, which operates in the cloud. Malekzadeh et al. [119] introduced Replacement AutoEncoder, a privacy-preserving sensing framework that transforms discriminative features corresponding to sensitive inferences into non-sensitive features to protect individuals' privacy. The possibility of recognizing the sensitive inferences and detecting the occurrence of them is eliminated, and is evaluated in an activity recognition task.

## 4.2 Motivation

### 4.2.1 The New Privacy Challenge

When addressing data privacy, the emphasis often lies on the confidentiality of the data per se, yet the implications of sensor fusion in multi-sensor environments are frequently overlooked. Although sensor fusion enhances application predictions through the integration of diverse data inputs—yielding significant benefits—this technique also raises substantial privacy concerns. In multi-sensor environments, the aggregation of data from various sensors can reveal intricate details about an individual's private life, including sensitive activities that are typically concealed. This duality poses a critical challenge: while the predictive utility of multi-sensor fusion is undeniable, it simultaneously has the capacity to expose highly private behaviors. As such, the development and deployment of these technologies must carefully balance the benefits against the potential for privacy invasion. Ensuring robust privacy safeguards and transparent policies in the deployment of sensor fusion technologies is crucial to maintaining trust and protecting individual privacy rights in increasingly monitored environments.

To demonstrate that this is a critically important and highly prevalent research question in practice, we conducted preliminary experiments to validate this issue. We collected datastreams from multi-sensor environments (in this case 2 office rooms) and tried to classify the activities by using these sensor data. Figure 4.2 shows an example of our sensor placement of the office setup. The sensor we use in the setup is AWAIR Omni<sup>®</sup> which collects indoor environmental quality (IEQ) factors data (e.g. CO<sub>2</sub>, PM2.5, illuminance, etc.), sound pressure level (SPL), humidity, and temperature. Characteristics of this sensor can be found in table 4.1. In this set of experiments, 5

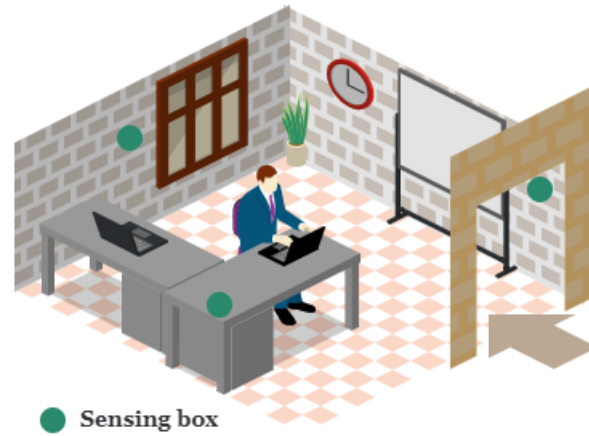
participants engaged in various activities typical of office environments, including: (1) typing on a computer, (2) conducting Zoom<sup>®</sup> calls, (3) writing on a whiteboard, and (4) standing while drinking. These activities were performed for 15 minutes each within two different offices where sensors were uniformly placed. Each participant underwent two sessions, each lasting 1 hour, during which sensors recorded data every 10 seconds. Following each participant's session, the room was vacated and left with open doors for 1 hour to allow for natural cleaning of the office environment. This precaution ensured that pollutants emitted during one session did not affect subsequent experiments. Thus, all experimental conditions started with identical indoor air pollutant concentrations.

We choose to perform the above four activities for 2 reasons. First, these are normal indoor office activities that we have every day. Second, typing on a computer, conducting Zoom<sup>®</sup> calls and writing on a whiteboard are often considered as very personal activities and highly privacy-sensitive, however, standing while drinking is a meaningful activity to help track the user's health (The intake of water in a standing position can cause damage to your kidneys) [120]. Ideally, we want the data streams can only detect standing while drinking, and can not detect the privacy-sensitive ones. However, we run a simple Support Vector Machines (SVM)[121], and we train the classification model in a self-training fashion. The reason we use self-training is that in supervised training, it's already well-known that given the sensor data and the correct labels, we can successfully classify the activities [122]–[124]. But in reality, when people share the sensor data, it's not always sent with the labels. So if the soft label from self-training can match the actual label in the datasets, it means even sending sensor data streams without labels, it's still a potential privacy leakage.

Figure 4.3 shows the confusion matrix of our preliminary study, where  $A_0$ ,  $A_1$ ,  $A_2$  and  $A_3$  reflect the four above activities. We found that the soft label of each activity from self-training can match the real label with high accuracy from 82% to 87%, which indicates that this activity privacy leakage in a multi-sensor environment is crucial and it's important and meaningful for us to come up with a private activity protection framework before sharing multi-sensor data streams.

### **4.2.2 Beneficial for IoT Data training among users**

For users owning IoT data, a prevalent challenge often revolves around the inadequacy of local data, which significantly impedes the ability to train effective machine learning models. As a result, there is a growing inclination towards collaborative learning approaches. Traditionally, two kinds of primary methodologies have emerged: (1) Cloud Computing based Methods [125], [126]: which



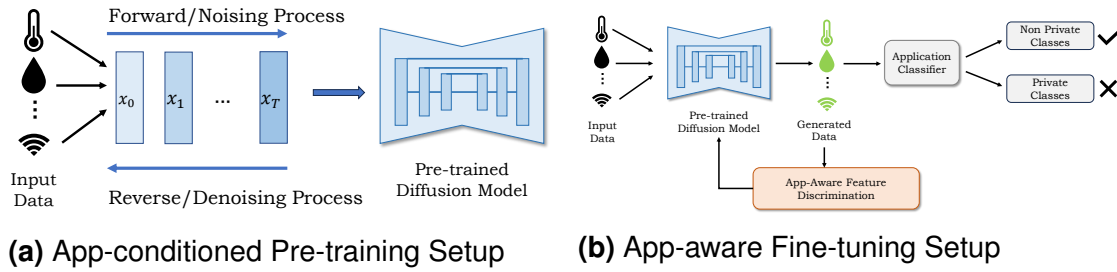
**Figure 4.2: Example of sensor data collection.**

**Table 4.1: Characteristics of sensors**

Device	Sensor	Value range	Error	Unit
Awair Omni	Humidity	[0, 100]	$\pm 2\%$	%
	Temperature	[-40, 125]	$\pm 2^\circ\text{C}$	$^\circ\text{C}$
	CO <sub>2</sub>	[400, 5000]	$\pm 75\text{ppm}$	ppm
	TVOC	[0, 60000]	$\pm 10\%$	ppb
	PM2.5 SPL	[0, 1000] [48, 90]	$\pm 15$	$\mu\text{g m}^{-3}$ dBA

A0	0.85	0.05	0.06	0.04
A1	0.03	0.84	0.05	0.08
A2	0.06	0.04	0.87	0.03
A3	0.07	0.05	0.06	0.82
	A0	A1	A2	A3

**Figure 4.3: Confusion matrix of our preliminary study**



**Figure 4.4: Training setups of PrivateHub**

involve aggregating user data within cloud data centers for collective machine learning training. While this approach can substantially enhance model performance, it introduces potential privacy risks due to the necessity of transmitting raw data to centralized cloud servers. (2) Federated Learning based Methods [69], [127]: which mitigate privacy concerns by solely uploading model parameters for aggregation, yet it remains vulnerable to privacy breaches such as model inversion attacks. Conversely, overly stringent measures to safeguard data privacy, including techniques like differential privacy [96], may inadvertently compromise the efficacy of machine learning models by limiting the amount of useful data available for training.

However, in practical contexts, the actual sensitivity of data privacy may not always align with conventional perceptions. For instance, recording environmental factors like ambient temperature and CO<sub>2</sub> levels typically does not classify as privacy-sensitive information. Similarly, many activities inferred from such sensor data are often not deemed sensitive from a privacy perspective. However, the primary apprehension arises when seemingly innocuous data points are leveraged to infer sensitive activities, thereby discouraging users from participating in collaborative data sharing initiatives. Hence, this paper proposes a novel data generation architecture designed to uphold robust privacy measures while ensuring that meaningful activities can still be effectively identified and analyzed. The framework aims to safeguard privacy-sensitive activities while enabling users to confidently share their data, thereby fostering greater collaboration and innovation in IoT-driven applications.

### 4.3 System Design

In this section, we illustrate the details of the proposed method. We draw our motivation from the fact that existing methods mainly focus on how to improve the efficiency of activity prediction, where the privacy issue among these methods are largely neglected, expecting effective techniques to discriminate privacy sensitive data and the non-private sensitive ones. Therefore, our method is motivated to leverage the powerful generative ability of diffusion model in discriminating the aforementioned data. Particularly, we propose a general paradigm to optimize the diffusion model in producing App-aware data, along with two stages, namely App-Conditioned pre-training (ACP) and App-Aware fine-tuning (AAF). Figure 4.4 shows the training setup of our method. Specifically, in Figure 4.4(a), ACP pre-trains a diffusion model on multi-sensor data datastreams with task category embedding, so that the model is capable of producing app-conditioned data after this stage. In this process, we aim to build a pre-trained diffusion model for multi-sensor data streams generation and use it as a base model for the AAF step. The steps of ACP follow the regular procedure of diffusion model training. We discuss the detailed steps in the following section 4.3.2.

Afterward, AAF enables the diffusion model to further discriminate privacy sensitive and non-sensitive data by conducting contrastive learning for optimization, where this stage adopts an off-the-shelf activity classifier to extract features from different generated data and encourages the model to produce data with closer feature distance to the privacy non-sensitive ones. Figure 4.4(b) shows the training setup for AAF. More specifically, once we have the pre-trained model from ACP, the users can input their multi-sensor data streams into the diffusion model to generate synthetic data. Then, the off-the-shelf activity classifier (we use a simple CNN structure in our method as an example, any off-the-shelf activity classifier method can replace the CNN classifier.) will classify the private and non-private classes. On the other hand, we design an App-aware feature discrimination function (AAFD), which integrates contrastive learning to distinguish the generated data. We aim to generate the multi-sensor data streams more related to non-private applications and less related to private applications. We achieve this by using the feature of real data and generated data to calculate the loss function and send it to update the diffusion model at each iteration. Moreover, we understand that the generated data may not be able to have the expected performance at all conditions. For example, in some iterations, the generated data can not perform application classification with desired accuracy. To eliminate the low-quality data generation, we introduce a data filtering process to ensure that only the data generated can achieve certain classification accuracy will be used to

send to AAFD. We discuss the detailed steps in the following section 4.3.3.

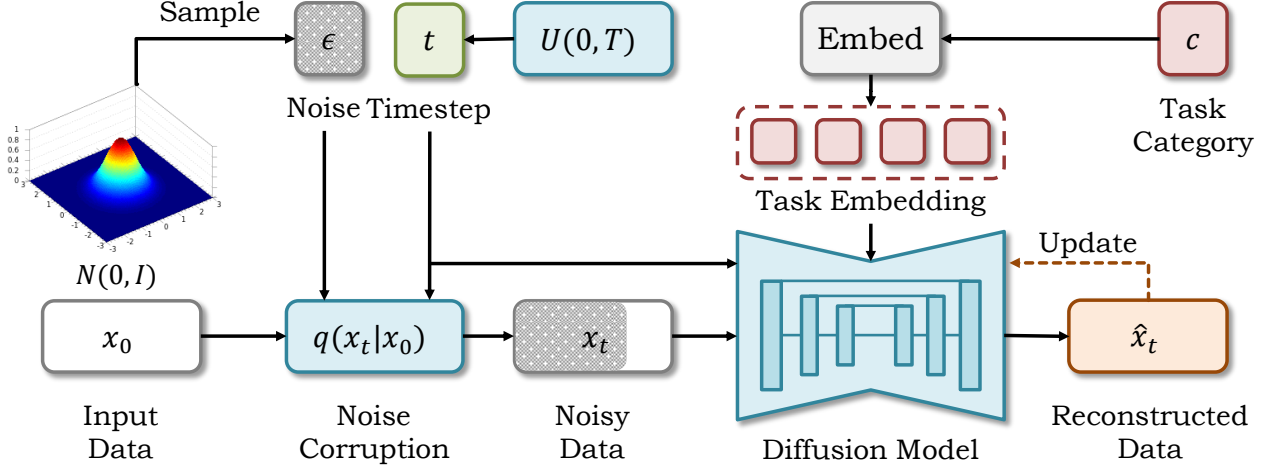
### 4.3.1 Design Fashion of Pre-training and Fine-tuning

We choose to adopt the pre-training and fine-tuning fashion to design our system for three reasons: **(1) Accuracy and Efficiency:** pre-training and fine-tuning architectures offer several distinct advantages over training models from scratch. Pretrained models leverage extensive datasets to learn rich and generalizable feature representations, significantly enhancing data efficiency. This foundational knowledge enables the models to achieve or surpass the performance of models trained from scratch in considerably shorter timeframes and with less data during the fine-tuning phase. However, directly training the model from the beginning to the end of the process will cause longer training time. On the other hand, the model will deal with noise corruption and updating the contrastive loss function at the same time, which lead to poor accuracy performance. **(2) Flexibility:** pre-trained models can be adapted across different applications with minimal adjustments, provided there is sufficient similarity between the applications. In reality of our multi-sensor application privacy challenge, it's highly possible that different users have varying standards for the privacy of their own data and activities. For instance, some users may not consider their dining activities to be privacy-sensitive, while others may strongly disagree. Therefore, the flexibility of data generation systems becomes critically important. By utilizing architectures that incorporate pre-training and fine-tuning, we can ensure that data tailored to the diverse privacy requirements of different users can be generated quickly and flexibly. In contrast, employing direct training would necessitate starting the training process from scratch for every unique user demand, which could lead to significant inconvenience.

### 4.3.2 App-Conditioned Pre-training

Since there is no available pre-trained models to support our motivation, the first step is to train a diffusion model with the multi-sensor data streams, i.e., the ACP stage. In doing so, ACP trains the diffusion model following the standard procedures, which consist of the **training** and **sampling** processes, where Figure 4.5 demonstrates the overall pipeline of the ACP stage.





**Figure 4.5: The overall pipeline of the App-Conditioned Pre-training (ACP) stage.**

**Training.** Given an input data  $x_0$ , we firstly randomly sample a noise  $\epsilon$  and a timestep  $t$  from the random Gaussian distribution  $N(0, I)$  and uniform distribution  $U(0, T)$ , respectively, where  $I$  denotes an identity diagonal matrix and  $T$  represents the maximum timestep. Then, we use the sampled  $\epsilon$  and  $t$  to corrupt the input clean data  $x_0$  into a series of noisy data, termed as  $\{x_1, x_2, \dots, x_T\}$ , where the process  $q(x_t|x_{t-1})$  is formulated by:

$$\begin{aligned}
 q(x_t|x_{t-1}) &= N(x_t; \sqrt{1 - \beta_t} \cdot x_{t-1}, \beta_t \cdot I), \\
 q(x_{1:T}|x_0) &= \prod_{t=1}^T q(x_t|x_{t-1}),
 \end{aligned} \tag{4.1}$$

where the aforementioned equations denote the reparameterization trick in DDPM [128]. Following Eq. 4.1, we can sample  $x_t$  at any timestep  $t$  with  $\alpha_t = 1 - \beta_t$  and  $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ , where  $x_t$  is computed through the following equation:

$$\begin{aligned}
 x_t &= q(x_t|x_0), \\
 &= N(x_t; \sqrt{\bar{\alpha}_t} \cdot x_0, (1 - \bar{\alpha}_t) \cdot I), \\
 &= \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon,
 \end{aligned} \tag{4.2}$$

where  $\sqrt{\bar{\alpha}_t}$  is a blending scalar correlated to the noise schedule of DDPM [128].

**Sampling.** The sampling process of diffusion model  $p_\theta(x_0 : T)$  starts from a random Gaussian noise  $x_T \sim N(0, I)$ , and iteratively de-noises from it into the final clean data  $x_0$  conditioned on the task embedding  $\mathbf{E}_c$ , where the sampling process is formulated as:

$$\begin{aligned} p_\theta(x_{t-1}|x_t) &= N(x_{t-1}; \mu_\theta(x_t, t), \Sigma), \\ p_\theta(x_{0:T}) &= p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t), \end{aligned} \quad (4.3)$$

Specifically, we use DDIM sampler during the sampling process, where the final clean data  $x_0$  is generated through:

$$\begin{aligned} x_{t-1} &= \sqrt{\bar{\alpha}_{t-1}} \cdot \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon_\theta(x_t, \mathbf{E}_c, t)}{\sqrt{\bar{\alpha}_t}}, \\ &+ \sqrt{1 - \bar{\alpha}_{t-1}} \cdot \epsilon_\theta(x_t, \mathbf{E}_c, t), \end{aligned} \quad (4.4)$$

### 4.3.2 Optimization Objective

In training, the usual variational bound on negative log-likelihood is optimized by:

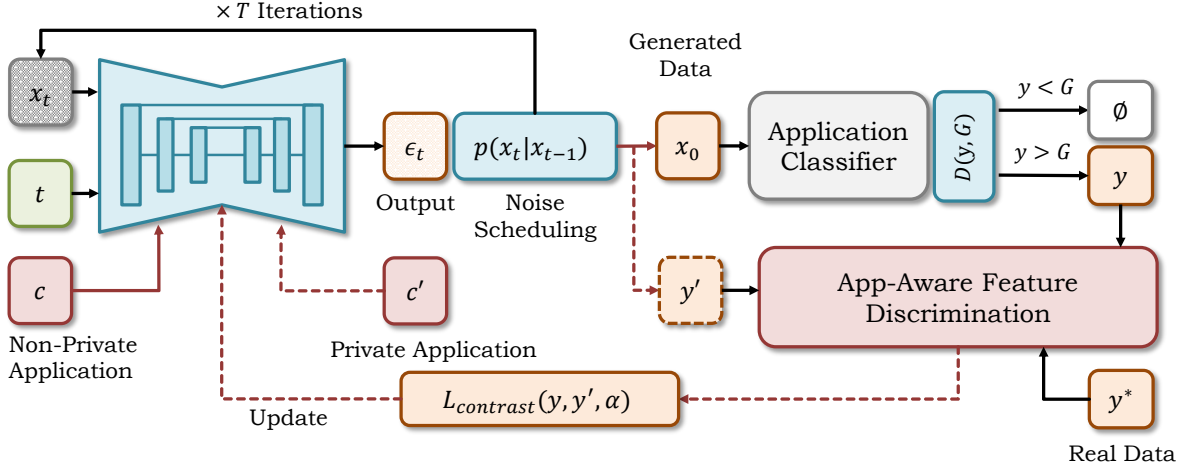
$$\begin{aligned} \mathbb{E}[-\log p_\theta(x_0)] &\leq \mathbb{E}_q \left[ -\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right] = \\ \mathbb{E}_q \left[ -\log p(x_T) - \sum \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right] &=: L, \end{aligned} \quad (4.5)$$

where  $L$  in the equation above is written as:

$$L = \mathbb{E}_q \left[ \begin{array}{c} D_{KL}(q(x_t|x_0)||p(x_T)) + \\ \sum_{t=1} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) \\ - \log p_\theta(x_0|x_1) \end{array} \right]. \quad (4.6)$$

Herein, KL refers to the Kullback-Leibler divergence.

To optimize the diffusion model, we use an embedding layer with a learnable matrix  $\mathbf{M}$  to project the task category label  $c$  into task embedding  $\mathbf{E}_c$ , where  $\mathbf{E}_c$  then serves as the condition for the diffusion model to de-noise  $x_t$ . Besides the task category label  $c$ , we also incorporate statistical features for each task to enrich the information available about each task. The features include the



**Figure 4.6: The overall pipeline of the App-Aware Fine-tuning (AAF) stage.**

mean, standard deviation, and Z-score, calculated as  $(x - \mu)/\sigma$ , where  $x$  is an observed value,  $\mu$  represents the mean, and  $\sigma$  denotes the standard deviation. Once  $x_t$  and  $\mathbf{E}_c$  are computed, we send  $x_t$ ,  $\mathbf{E}_c$ , and  $t$  into the diffusion model to predict noise, and minimize the Mean-Squared Error (MSE) distance between the predicted and the random Gaussian noise  $\epsilon$ , where the loss function  $L(\theta)$  to optimization the model parameters  $\theta$  of the diffusion model is written as:

$$L(\theta) = \|\epsilon - \epsilon_\theta(x_t, t, \mathbf{M}_c)\|^2, \quad (4.7)$$

where we follow DDPM in predicting noise  $\epsilon_\theta(x_t, t, \mathbf{M}_c)$  instead of the mean, so as to fit the data distribution. Herein, we conduct the diffusion model following the original architecture of DDPM [128], which uses an enhanced U-net architecture that consists of an encoder, a decoder, and multiple Transformer [129] blocks. Particularly in DDPM, the encoder mainly down-samples the input image into latent representations; the decoder up-samples the latent representations to the original size of image; the Transformer blocks learn the internal correlation between image patches via the self-attention mechanism. In the setting of multi-sensor environment, the U-net firstly encodes the noisy data  $x_t$  into latent representations, and decodes the representations into predicted noise  $\epsilon_\theta(x_t, t, \mathbf{M}_c)$ , where the Transformer blocks establish the relationship of data at various sensor time. In this way, we update the model parameters  $\theta$  of the diffusion model until convergence, where the optimized diffusion model is capable of producing data according to task categories.

### 4.3.3 App-Aware Fine-tuning

With the assistance of the ACP stage, the diffusion model is able to produce realistic data according to various tasks. Nevertheless, the intrinsic capabilities of the trained model still fails to discriminate privacy sensitive and non-sensitive data. To improve the inefficient capabilities, tailored optimization for data discrimination is expected, recent studies show that Reinforcement Learning with Human Feedback (RLHF) [130], [131] has demonstrated superior proficiency in aligning model outputs with human preferences. However, reinforcement learning algorithms, especially those based on value functions like Q-learning [132], frequently encounter stability and convergence issues during training. These arise from their dependency on accurate value estimates, which are influenced by policy changes, leading to high variance. Also, designing effective reward functions that reflect long-term goals is challenging and critical, as it significantly impacts learning outcomes. Additionally, the performance of these algorithms heavily relies on the complexity and observability of their environments, which can impede learning effective strategies. Due to the need for extensive data and prolonged training, these algorithms also require significant computational resources, limiting their applicability in resource-constrained scenarios. Moreover, although reinforcement learning can perform well in specific tasks, the strategies learned often struggle to generalize to different tasks or slightly altered environments.

Thus, we draw another insight from the advancement of contrastive learning, which digs positive and negative samples in a self-supervised manner, and enables the model to generate outputs that are more similar to the positive samples. This technique has illustrated promising performance in a wide series of down-stream tasks in the computer vision community, and shows superior efficiency compared to RLHF without the extra requirements of human feedback. Figure 4.6 shows the overall pipeline of our AAF stage. To enhance the diffusion model with contrastive learning, we need to consider from two perspectives: (1) the rules of selecting positive and negative samples; (2) the loss function of contrastive learning. In the following texts, we present the details of both aforementioned perspectives.

**Rules to Select Positive and Negative Samples.** Our goal of contrastive learning is to enable the diffusion model to discriminate privacy sensitive and non-sensitive data. Therefore, the rules to select positive and negative samples are straightforward, where we choose the real non-sensitive data as positive sample, and treat generated data with respect to privacy sensitive task categories. In

details, given the privacy sensitive task category  $c'$  and non-sensitive one  $c$ , we follow the standard process of Eq. 4.4 and generate the corresponding data with the pre-trained diffusion model, termed as  $x_0$  and  $y'$ , respectively. Then, we randomly sample a real non-sensitive data  $y^*$  from the multi-sensor data streams  $\mathcal{D} = \{y_1^*, y_2^*, \dots, y_{N_D}^*\}$  with the total length of dataset termed as  $N_D$ . In this way, we leverage  $y^*$  and  $y'$  as the positive and negative samples, respectively, where  $x_0$  and  $y'$  are then used in later processes of the AAF stage.

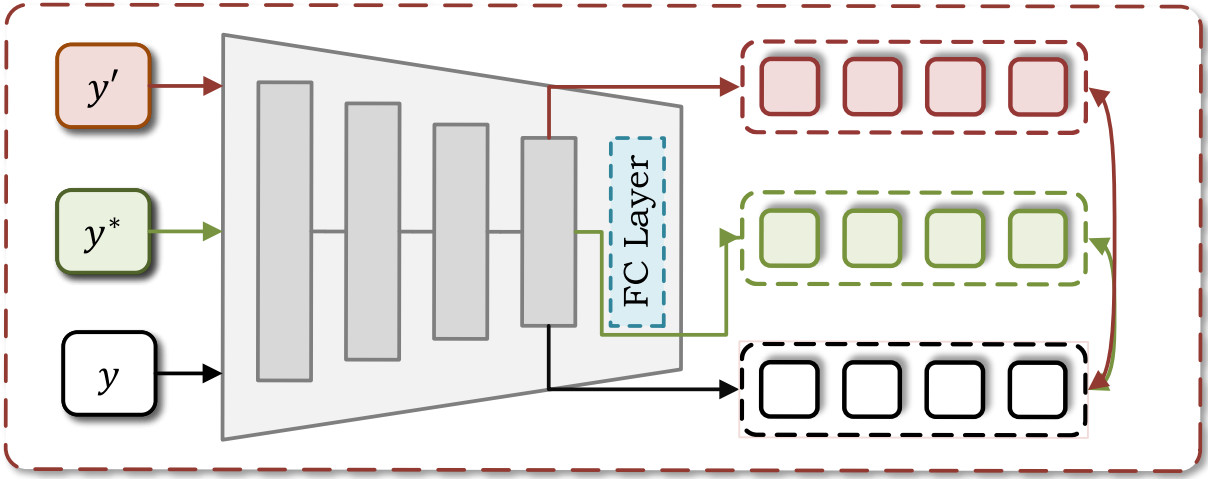
**Data Filtering.** Before we compute the loss function with contrastive learning, it is vital to ensure the data quality of  $x_0$ . Once we optimize the diffusion model with ill-presenting quality of  $x_0$ , there are possibilities that  $x_0$  has closer distance to  $y'$  than  $y^*$ , where might lead to inferior optimization during fine-tuning. To solve this problem, we leverage an off-the-shelf activity classifier  $\mathcal{E}_\phi$  with its model parameters as  $\phi$ . We observe that low-quality data would cause  $\mathcal{E}_\phi$  to produce inaccurate results, where we set a threshold  $G$  to filter the produced low-quality data, where the data filtering process  $D(x_0, G)$  is formulated as:

$$y = D(x_0, G) = \begin{cases} x_0, & \mathcal{E}_\phi(x_0) > G \\ \emptyset, & \mathcal{E}_\phi(x_0) \leq G \end{cases} \quad (4.8)$$

Afterward, we use the filtered data  $y$  to compute the loss function of contrastive learning. If the data filtering process outputs  $\emptyset$ , we do not compute the loss function of contrastive learning in this training iteration. In our experiment, we use a simple convolutional neural network (CNN) for application classification, which contains four convolutional layers followed by a ReLU activation function with 2 fully connected layers.

### 4.3.3 App-Aware Feature Discrimination

Once  $y$ ,  $y'$ , and  $y^*$  are ready, the final step is to optimize the diffusion model with the contrastive learning, i.e., the App-Aware Feature Discrimination (AAFD) process. Figure 4.7 shows the overall pipeline of AAFD, where the model in gray represents the off-the-shelf activity classifier. To perform AAFD, we use the activity classifier  $\mathcal{E}_\phi$  as a feature extractor, where we leverage the latent representations before the final Fully-Connected (FC) layer as extracted features, and then conduct the contrastive learning by comparing various features of different data, encouraging the diffusion model to generate outputs that are more similar to  $y^*$  rather than  $y'$ . Specifically, we firstly input  $y$ ,  $y'$ ,



**Figure 4.7: The overall pipeline of the App-Aware Feature Discrimination (AAFD). Herein, the model in gray represents the off-the-shelf activity classifier; “FC” denotes the fully-connected layer, which is normally a single linear projection layer to project the final latent representations into a probability distribution.**

and  $y^*$  into  $\mathcal{E}_\phi$ , and obtain their corresponding features, termed as  $\mathbf{F}$ ,  $\mathbf{F}'$ , and  $\mathbf{F}^*$ , respectively. Then, we compute the loss function of contrastive learning (termed  $L_{contrast}(\theta)$ ) following the standard form of triplet loss, which is written as the following equation:

$$L_{contrast}(\theta) = [\|\mathbf{F} - \mathbf{F}^*\|^2 - \|\mathbf{F} - \mathbf{F}'\|^2 + \alpha], \quad (4.9)$$

where  $\alpha$  represents a hyper-parameter that determines the margin value of the triplet loss. In training, we optimize the diffusion model with the loss function in Eq. 4.9 until convergence. For further analyses of this stage, the contrastive learning encourages the diffusion model to produce data  $y$  that have closer feature distance to the real data  $y^*$  rather than  $y'$ , and allow the activity classifier  $\mathcal{E}_\phi$  to output more accurate results with the corresponding feature inputs. In this way, we are capable of allowing the diffusion model to producing privacy non-sensitive data, meanwhile maintaining the sample quality in its generated data.

## 4.4 Evaluation

To evaluate the performance of `PrivateHub` on generality, practicability and usefulness, we experiment our method on real-world smart home and smart office multi-sensor datasets on the following research questions:

- **RQ1:** How does `PrivateHub` perform on synthetic data generation? Can the generated data streams from `PrivateHub` detect non-private applications and conceal private ones? And what are the performance differences between `PrivateHub` and the other methods?
- **RQ2:** How does `PrivateHub` perform if the users switch the selection of the private/non-private applications? Will it influence the overall performance? This is a practical test to see if `PrivateHub` is robust enough to handle different users.
- **RQ3:** How do the generated data streams perform on their original purposes? (e.g., CO<sub>2</sub> prediction, temperature prediction, etc)
- **RQ4:** How does `PrivateHub` perform if there are missing application labels?

### 4.4.1 Applications and Datasets

For the new application privacy challenge in multi-sensor environments, we opt to validate our approach using smart home and smart office datasets. It is important to note that this does not imply that our method is limited to data generation in these two scenarios. We selected these scenarios because they represent common examples of multi-sensor environments.

**(1) Smart Home Activities:** The CASAS dataset<sup>1</sup>, a product of the CASAS smart home project, serves as a resource for recognizing daily living activities (ADL) through sequences of sensor states over time, aiming to facilitate independent living applications. The data collection took place across different apartments, each outfitted with a trio of sensor types: motion, temperature, light, water, burner and door sensors. We keep the same data pre-processing mechanism as mentioned in [88]. We have chosen five specific datasets named “Milan”, “Kyoto1”, “Kyoto2”, “Kyoto3” and “Kyoto4” selected for the consistency in their sensor data representation. Within each dataset, we

---

<sup>1</sup><https://casas.wsu.edu/datasets/>

have streamlined the original ADL categories into 11 home activity-related categories such as “sleep”, “eat”, “leave home”, “enter home”, “bath”, etc. Each data entry is a categorical time series with a length of 2,000, depicting sensor states across a certain time span.

**(2) Smart Office Activities:** We use the above mentioned self-collected smart office dataset to illustrate the smart office activities. Figure 4.2 shows an example of our sensor placement of the office setup. We follow the same setup as mentioned in our motivation section.

#### 4.4.2 Experimental Setup and Evaluation Metrics

We implemented our experiments using one NVIDIA RTX 3090 GPU. For the diffusion model settings, the step size of the forward diffusion process is controlled by a variance schedule  $\beta_t \in (0.0001, 0.05)$ , where  $t$  ranges from 1 to  $T$ . The maximum diffusion step is set to be 100. The batch size was 128, and training spanned 200 epochs with early stopping using a patience of 30 epochs to counter overfitting.

For the CASAS datasets, since we are under the pre-training and fine-tuning setup, we combine these datasets together for training in a leave-one-out fashion. For example, If we assume “Kyoto4” is a new user, we use “Milan”, “Cairo”, “Kyoto1”, “Kyoto2” and “Kyoto3” to perform the pre-training step, then only use “Kyoto4” for fine-tuning and testing. We divided the leave-one-out dataset into fine-tuning training and testing sets using an 80-20 split.

For our self-collected smart office dataset, since we collected the multi-sensor data streams on 2 office rooms, we chose to pre-train on one room, and fine-tuning on other room, and we also divided the fine-tuning room datasets into fine-tuning training and testing sets using an 80-20 split.

For comparison methods, since PrivateHub is a synthetic data generation framework, we compare our method within the same functionality. However, it’s crucial to state that most of the GAN-based models [106], [133], [134] have shown limitations that the generated data often lack diversity and pose significant challenges during the training phase due to mode collapse and unstable convergence. So we choose the following GAN-based methods tailored to our cases. We compare PrivateHub with three different types of methods.

- **TimeGAN [107]:** Different from the other GAN-based method, TimeGAN has shown good results to be trained stably and successfully generating time series data. For TimeGAN, we adopt our pre-training and fine-tuning setup, which means that we replace the diffusion model



part solely with the TimeGAN framework and compare the results.

- **Conditional GAN [135]:** Conditional GAN extends the original framework by incorporating conditional information, enabling the generation of data samples that are conditioned on specific attributes or labels. This model architecture enhances the versatility and directed capability of GANs, making them more suitable for tasks that require controlled output generation. In our case, the conditional is to generate data that can conceal private activities and still maintain reasonable accuracy on non-private ones. So we assign the discriminator with a lower score on the private activities and a higher score on the non-private ones. Since this method needs to put the condition features within the training process, we only use direct training for Conditional GAN.
- **Occupancy-GAN [92]:** Occupancy-GAN is designed to conditionally generate data as well. Different from Conditional GAN, in Occupancy-GAN, the generated data is classified by two application classifiers. Two classifiers output 2 different losses and add these losses into the generator. It utilized a customized loss combination function to guide the generator by reducing loss from GAN and operational classifier ( $C_1$ ) and increasing loss from expositional classifier ( $C_2$ ). It eventually tries to optimize the  $C_1$  and break down the  $C_2$ . Since this method needs to put the condition features within the training process, we only use direct training for Occupancy-GAN.

For evaluation metrics, we use accuracy for the classification tasks. We evaluate the performance of each user’s test data. We repeated each experiment 5 times and averaged results for classification accuracy. For CASAS datasets, we label the ”bath” and ”leave home” as private activities, the rest we assume them as non-private ones. For smart office dataset, we label conducting Zoom<sup>®</sup> calls as the private ones, the rest we assume them as non-private ones. Please note that this is our default setting, in RQ2 we switch the application selections.

### 4.4.3 Results

#### 4.4.3 RQ1: Performance of synthetic data generation

In this section, we compare how PrivateHub performs on synthetic data generation with other methods of concealing private activities. Table 4.2 shows that PrivateHub can achieve com-

Table 4.2: Performance Metrics Comparison

Dataset	Activity Acc	TimeGAN [107]	Conditional GAN [135]	Occupancy-GAN [92]	PrivateHub	Raw data
Milan	Non-private	0.774 ± 0.052	0.653 ± 0.093	0.692 ± 0.032	0.840 ± 0.067	0.84
	Bath	0.425 ± 0.045	0.498 ± 0.025	0.512 ± 0.020	0.380 ± 0.019	0.78
	Leave home	0.471 ± 0.021	0.495 ± 0.035	0.510 ± 0.022	0.393 ± 0.023	0.83
Kyoto1	Non-private	0.686 ± 0.038	0.622 ± 0.050	0.539 ± 0.046	0.778 ± 0.064	0.80
	Bath	0.421 ± 0.043	0.494 ± 0.034	0.502 ± 0.022	0.365 ± 0.016	0.72
	Leave home	0.441 ± 0.037	0.492 ± 0.019	0.500 ± 0.015	0.381 ± 0.023	0.81
Kyoto2	Non-private	0.760 ± 0.027	0.635 ± 0.026	0.663 ± 0.019	0.842 ± 0.023	0.83
	Bath	0.415 ± 0.039	0.497 ± 0.029	0.510 ± 0.024	0.286 ± 0.068	0.74
	Leave home	0.438 ± 0.033	0.490 ± 0.021	0.495 ± 0.016	0.392 ± 0.033	0.87
Kyoto3	Non-private	0.688 ± 0.041	0.612 ± 0.025	0.533 ± 0.039	0.725 ± 0.025	0.84
	Bath	0.410 ± 0.045	0.496 ± 0.031	0.507 ± 0.025	0.239 ± 0.097	0.70
	Leave home	0.432 ± 0.030	0.487 ± 0.023	0.499 ± 0.017	0.301 ± 0.014	0.78
Kyoto4	Non-private	0.767 ± 0.043	0.627 ± 0.041	0.683 ± 0.034	0.841 ± 0.030	0.83
	Bath	0.418 ± 0.042	0.491 ± 0.028	0.508 ± 0.020	0.376 ± 0.055	0.76
	Leave home	0.435 ± 0.036	0.484 ± 0.018	0.503 ± 0.014	0.345 ± 0.026	0.80
Smart office	Non-private	0.812 ± 0.022	0.721 ± 0.026	0.821 ± 0.031	0.953 ± 0.027	0.95
	Zoom call	0.421 ± 0.029	0.545 ± 0.034	0.428 ± 0.037	0.396 ± 0.046	0.93

parable accuracy performance compared to raw data on non-private activities, and also achieve lower than 40% of private activities, which is around 40-50% lower than the raw data. We can also observe that, TimeGAN has better performance on both sides comparing to conditional GAN and Occupancy-GAN. However, comparing to PrivateHub, the non-private accuracy performance is significantly lower. Also, the private activities are also higher than PrivateHub, but we do observe that all methods have a lower private activity accuracy compared to raw data. The reason is that, for PrivateHub and TimeGAN, the pre-training and fine-tuning setup can help smooth the whole training process, which makes the generated data more similar to the raw data distribution on the non-private activity features. Also, our designed AAFD can better capture the difference between private and non-private activities, which makes the accuracy on private activities much lower.

On the other hand, PrivateHub outperforms TimeGAN. The reason is that, although we only change the diffusion model with TimeGAN, the GAN architecture is not really suitable for this task compared to diffusion models. Diffusion models are renowned for their ability to generate exceptionally high-quality samples, characterized by impressive detail and coherence, due to their incremental denoising approach that methodically removes noise throughout the data generation process. This attribute ensures that the produced samples closely mimic real data. Furthermore, these models exhibit superior training stability, attributed to their probabilistic foundations based on Markov chains and gradual gradients. Such stability mitigates common training issues like convergence difficulties, providing a smoother learning curve. Additionally, the training process of diffusion models is particularly intuitive, involving a direct and clear engagement with both the data and noise distributions. This process is facilitated by progressively refining noisy data through a denoising process, making the models both effective and accessible for generating new data instances.

#### *4.4.3 RQ2: Performance of synthetic data generation with switched private and non-private activities*

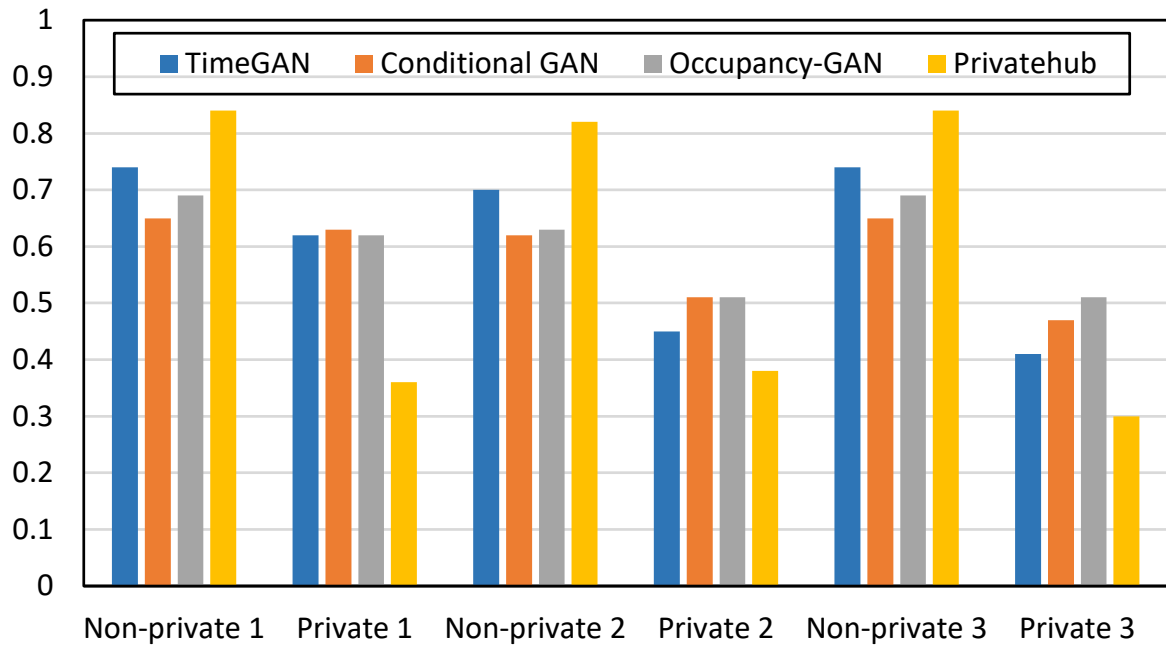
We compare the robustness of PrivateHub by switching the private and non-private activity labels. By default, we use "bath" and "leave home" as private activities, and the rest we label as non-private. However, different users may have different privacy activities. In this experiment, we switch the labels 3 times and we demonstrate the performance on "Milan" and "Kyoto1" datasets.

Private 1 we labeled "cook" and "eat" as private, Private 2 we labeled "personal hygiene" and "eat" as private, Private 3 we kept the "bath" and "leave home" to see the difference.

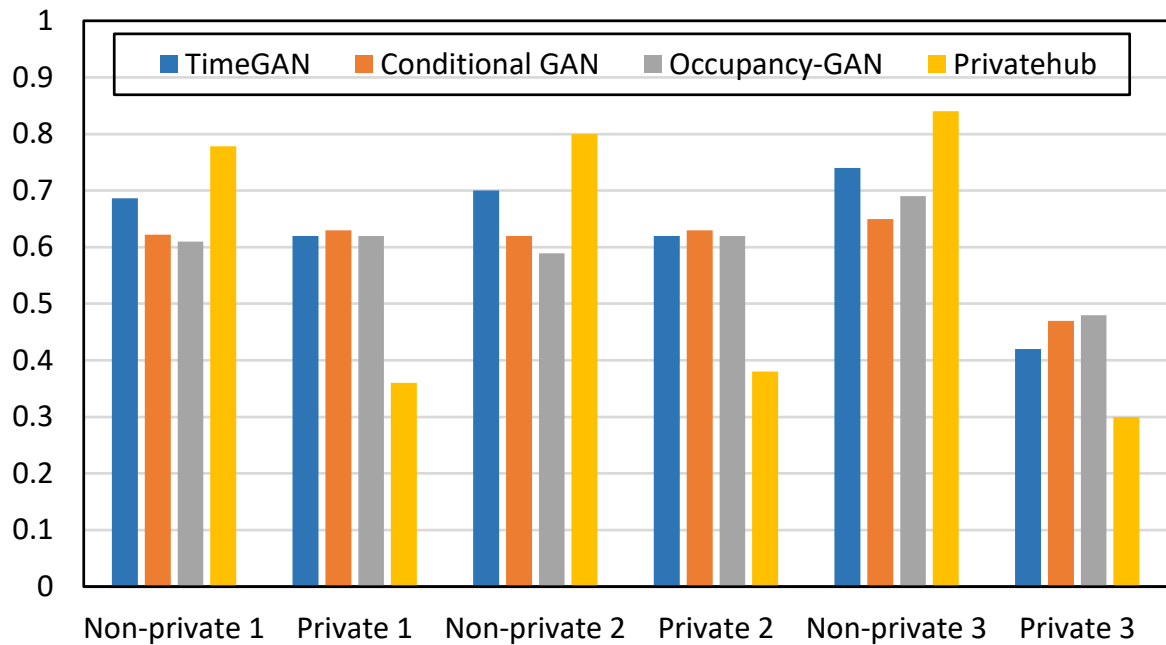
Figure 4.8 shows the result of private and non-private activity switch. We observe that the overall trend follows the same as RQ1. However, we see that for Milan, under Private 1 settings, the GAN-based methods can not generate qualified data streams. And for Kyoto1, both Private 1 and 2 settings can not generate qualified data, while `PrivateHub` can generate synthetic data under all 3 settings. The reason is that, compared to Diffusion Models, GAN-based models have limitations that restrict their effectiveness in applications that require outputs with high diversity. Although Conditional GANs offer a method to introduce conditional variables to the generator, controlling the specific types of outputs generated by GANs in practice still presents challenges. In contrast, Diffusion Models facilitate a more natural control over the outputs by adjusting the noise levels throughout the generation process, allowing for a more nuanced and flexible manipulation of the generated data. On the other hand, instead of using conditional embedding, we use contrastive learning based loss function, AAFD, to further fine-tune the generated data. Contrastive learning effectively learns data representations by maximizing the consistency between similar samples and minimizing the consistency between dissimilar samples, which helps the diffusion model to generate more non-private data and less private data. This approach enables the extraction of informative and discriminative features, which enhances the performance of subsequent tasks.

#### 4.4.3 RQ3: Performance of synthetic data generation with original purpose

In this section, we want to figure out if the generated data will damage the original purpose of each data stream. For example, in our collected data, the original purpose for `AWAIR Omni`<sup>®</sup> which collects indoor environmental quality (IEQ) factors data (e.g. CO<sub>2</sub>, PM2.5, illuminance, temperature, etc.) is to monitor the indoor environment of the smart office, and further perform prediction tasks, such as CO<sub>2</sub> level prediction, temperature prediction. There is a chance that the generated data may damage such data utility. Figure 4.9 shows the performance on CO<sub>2</sub> and temperature prediction. We use a simple LSTM [136] for time series prediction. We observe that all methods can achieve reasonable predictions and `PrivateHub` has the closet performance comparing to raw data.

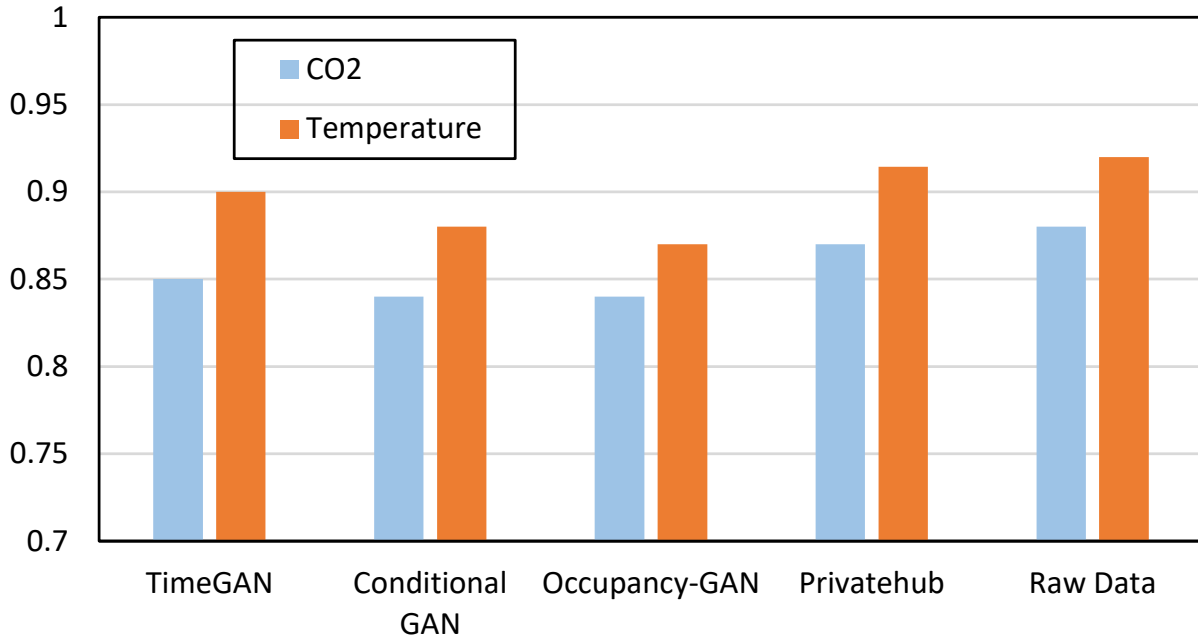


(a) Milan



(b) Kyoto1

**Figure 4.8: Performance comparison on switch private and non-private activities (Y-axis is accuracy)**



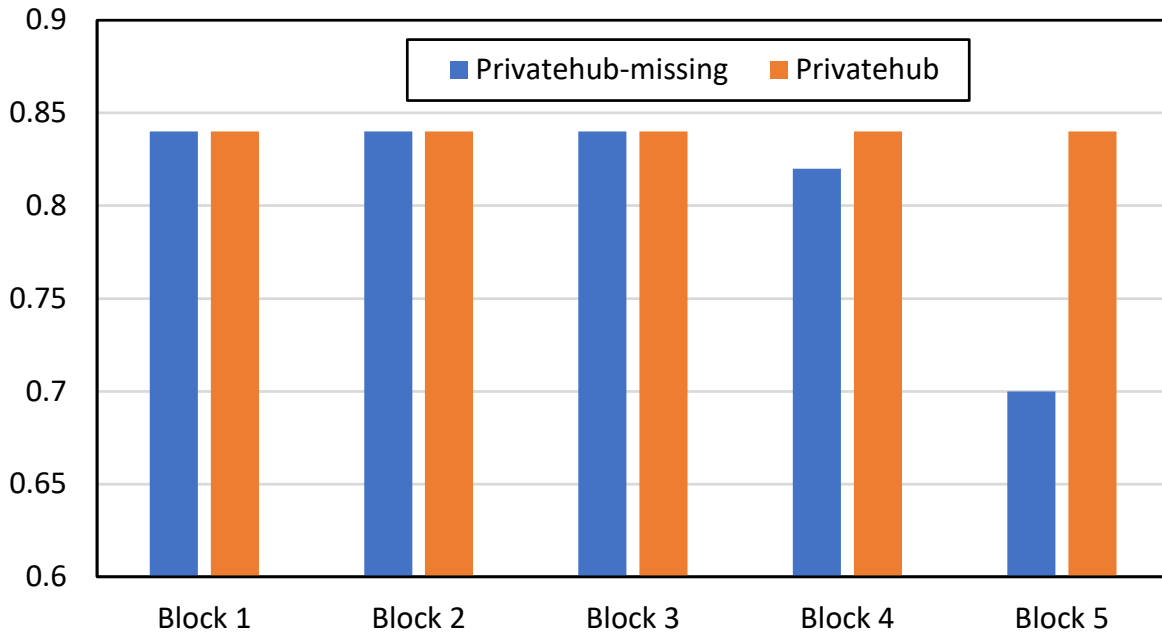
**Figure 4.9: Performance comparison on CO<sub>2</sub> and temperature prediction (Y-axis is accuracy)**

#### 4.4.3 RQ4: Performance of PrivateHub on missing application labels

In reality, it's very likely that when the user sends the data streams, there will be missing activity labels. However, it's crucial to understand whether the generated data from PrivateHub with partial activity labels can still be classified. In this experiment, we block 2 activity labels and train PrivateHub with all other labels then test if we can still identify the blocking 2 applications. We tried 5 different setups and compared PrivateHub with the missing label training version. We observe that for most (60%) of the setups, we have no accuracy lost. For 20% of the setups, we have around 2% accuracy drop. For the last setup, we have around a 14.2% accuracy drop. The reason is probably that the blocking label is highly related to private applications.

## 4.5 Conclusion

Multi-sensor environments are widespread, constantly collecting environmental data which enhances daily life but also introduces privacy risks in multi-sensor contexts. Our method utilizes contrastive learning within a diffusion model to generate synthetic data streams tailored for privacy in multi-



**Figure 4.10: Performance comparison on missing application labels (Y-axis is accuracy)**

sensor settings. We propose App-Conditioned Pre-training (ACP) and APP-Aware Fine-tuning (AAF). ACP conditions the model on multi-sensor data with application-specific embeddings, while AAF fine-tunes the model's ability to distinguish between privacy-sensitive and non-sensitive data. Our tests show that we effectively reduce privacy risks, lowering the detection accuracy of private applications by 40% to 50% and maintain the same level of performance on non-private applications compared to raw data.

# Chapter 5

## CONCLUSION

The proliferation of the Internet of Things (IoT) is integrating millions of new smart devices, including smartphones, sensors, and wearables, into our physical environment, significantly enhancing human life quality. The exponential increase in smart devices and users generates vast amounts of IoT sensing data, which, coupled with advances in deep learning, offers substantial potential for smart IoT applications. To realize these applications, users typically need to either (1) aggregate data from multiple users to construct robust models or (2) employ various types of sensors. However, such data aggregation raises significant privacy concerns. In multi-user environments, IoT edge devices often collect limited data, which is inadequate for training advanced deep learning models. Collaborative training approaches like cloud computing and federated learning are employed to develop robust models by pooling data from numerous users, but these techniques can compromise data privacy (e.g., through untrusted central servers or model inversion). In multi-sensor environments, sensor fusion involving various devices like lights, doors, speakers, and CO<sub>2</sub> sensors enriches our living and work environments, however this fusion can inadvertently expose private activities. In our previous research [4], we demonstrated that combining data from CO<sub>2</sub> and humidity sensors could accurately determine room occupancy. The challenge in multi-sensor environments is maintaining sensor accuracy for their intended purposes while avoiding the detection of sensitive activities.

In response to these concerns, this dissertation explores three strategies to achieve privacy-centric IoT applications without compromising the accuracy relative to non-privacy-preserving methods. (1) Personalized Federated Deep Reinforcement Learning (PFDRL), a system designed to aid local users in achieving private and efficient energy management. PFDRL adopts a decentralized federated learning (DFL) framework, eliminating the need for a central server and implementing personalized federated reinforcement learning to address standby energy reduction in residential buildings. (2) *Atlas*, a framework that ensures privacy and accuracy in IoT applications through personalized



federated local differential privacy (LDP). Initially, we establish a layer-sharing strategy known as the layer importance mask, which differentiates between global and personalized layers within the local model. Subsequently, we incorporate a weighted LDP mechanism, applying noise to the global layers prior to their transmission to the federated learning framework for aggregation. Lastly, we merge the local personalized layers with the aggregated global layers to execute IoT tasks. (3) `PrivateHub`, an innovative method for generating synthetic data streams that employs contrastive learning within a diffusion model framework. This method conditionally generates data streams for private multi-sensor scenarios, effectively differentiating non-private applications while protecting the privacy of sensitive ones. `PrivateHub` harnesses the robust generative capabilities of diffusion models to accurately identify various data types. `PrivateHub` includes a structured framework that enhances the performance of diffusion models for generating application-specific data, consisting of two main phases: App-Conditioned Pre-training (ACP) and APP-Aware Fine-tuning (AAF). In the ACP phase, the diffusion model undergoes pre-training using standard techniques on multi-sensor data streams to produce application-conditioned data. Following this, the AAF phase enhances the model’s ability to distinguish between privacy-sensitive and non-sensitive data using contrastive learning optimization. During this stage, we utilize a readily available application classifier to analyze features from the generated data, prompting the model to produce outputs whose feature distances align more closely with those of non-sensitive data.

### 5.0.1 Limitations and Future Work

While this dissertation has brought major improvements in the trade-off in multi-user and multi-sensor IoT data between accuracy and privacy, the presented approaches still have some limitations, which are worth exploring in the future. In this section, we review the limitations of our current designs and highlight some future research avenues as a continued effort toward private and accurate IoT applications.

**Communication efficiency optimization:** In realistic IoT settings, communication efficiency is a crucial factor to evaluate whether the designed system can be used in real-time. In this dissertation, we understand that our proposed method achieves lower communication cost simply because of the selected layer mechanism, which means we transmit less package to the central server and is not optimal for communication efficiency compared to other FL frameworks that are mainly focusing on communication reduction [67].

**Secure environment:** This dissertation mainly focuses on the trade-off between IoT data privacy and accuracy. It's necessary to highlight that, security is another important factor in a realistic IoT setup. In this dissertation, we did not pay much attention to the security side. In the future study, to achieve an accurate and private IoT framework, we should consider integrating security procedures to enhance our system.

**Hierarchical federated learning:** Our federated learning methods are based on single-layer FL settings, meaning we only have clients and a central server. In reality, IoT platforms are not always single-layer. For example, it's practical that in a smart building, the data streams are collected from low computational power devices and they connect with an edge gateway, then the gateways connect to the cloud server. Such a hierarchical setting brings more challenges on privacy, accuracy and communication efficiency. We aim to dive deeper into more realistic IoT settings and better optimize the trade-off between these aspects.

## REFERENCES

- [1] “Help net security.,” <https://www.helpnetsecurity.com>, 2021.
- [2] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, “Blockchain and federated learning for privacy-preserved data sharing in industrial iot,” *IEEE Transactions on Industrial Informatics*, 2019.
- [3] D. Marikyan, S. Papagiannidis, and E. Alamanos, “A systematic review of the smart home literature: A user perspective,” *Technological Forecasting and Social Change*, vol. 138, pp. 139–154, 2019.
- [4] M. F. R. M. Billah, N. Saoda, J. Gao, and B. Campbell, “Ble can see: A reinforcement learning approach for rf-based indoor occupancy detection,” in *Proceedings of the 20th international conference on information processing in sensor networks (co-located with CPS-IoT Week 2021)*, 2021, pp. 132–147.
- [5] I. Ayres, S. Raseman, and A. Shih, “Evidence from two large field experiments that peer comparison feedback can reduce residential energy usage,” *The Journal of Law, Economics, and Organization*, vol. 29, no. 5, pp. 992–1022, 2013.
- [6] O. Parson, G. Fisher, A. Hersey, *et al.*, “Dataport and nilmtk: A building data set designed for non-intrusive load monitoring,” in *2015 ieee global conference on signal and information processing (globalsip)*, IEEE, 2015, pp. 210–214.
- [7] M. Lu and J. Lai, “Review on carbon emissions of commercial buildings,” *Renewable and Sustainable Energy Reviews*, 2020.
- [8] “Berkeley lab standby power.,” 2021.
- [9] W. Wang, J. Su, Z. Hicks, and B. Campbell, “The standby energy of smart devices: Problems, progress, & potential,” in *2020 IEEE/ACM Fifth IoTDI*, IEEE, 2020.
- [10] L. B. Laboratory, <https://standby.lbl.gov/docs/>, 2008.

- [11] W. Wang, V. A. L. Sobral, M. F. R. M. Billah, N. Saoda, N. Nasir, and B. Campbell, "Low power but high energy: The looming costs of billions of smart devices,"
- [12] A. Dhungana, "Peer-to-peer energy sharing in mobile networks: Applications, challenges, and open problems," *Ad Hoc Networks*, 2020.
- [13] K. Gram-Hanssen, "Standby consumption in households analyzed with a practice theory approach," *Journal of Industrial Ecology*, 2010.
- [14] Y. Hu and D. Niu, "Fdml: A collaborative machine learning framework for distributed features," in *Proceedings of the 25th SIGKDD*, 2019.
- [15] M. Al Faruque and K. Vata, "Energy management-as-a-service over fog computing platform," *IEEE internet of things journal*, 2015.
- [16] M. Soliman and T. Abiodun, "Smart home: Integrating internet of things with web services and cloud computing," in *2013 IEEE 5th international conference on cloud computing technology and science*, 2013.
- [17] U. M. Aivodji, S. Gambs, and A. Martin, "Iotfla: A secured and privacy-preserving smart home architecture implementing federated learning," in *2019 IEEE Security and Privacy Workshops (SPW)*, 2019.
- [18] A. Taik and S. Cherkaoui, "Electrical load forecasting using edge computing and federated learning," in *2020 IEEE International Conference on Communications (ICC)*, 2020.
- [19] *Dataport Database*, <https://dataport.pecanstreet.org>, 2014.
- [20] W. Kong, "Short-term residential load forecasting based on lstm recurrent neural network," *IEEE Transactions on Smart Grid*, 2017.
- [21] G. Din, A. Mauthe, and A. Marnerides, "Appliance-level short-term load forecasting using deep neural networks," in *2018 International Conference on Computing, Networking and Communications*, 2018.
- [22] X. Luo, L. Oyedele, and A. Ajayi, "Development of an iot-based big data platform for day-ahead prediction of building heating and cooling demands," *Advanced Engineering Informatics*, 2019.

- [23] L. Yang, X. Chen, J. Zhang, and H. Poor, "Optimal privacy-preserving energy management for smart meters," in *IEEE INFOCOM*, 2014.
- [24] H. Chang, W. Chiu, H. Sun, and C. Chen, "User-centric multiobjective approach to privacy preservation and energy cost minimization in smart home," *IEEE Systems Journal*, 2018.
- [25] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, and C. Makaya, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, 2019.
- [26] S. Lee, "Federated reinforcement learning for energy management of multiple smart homes with distributed energy resources," 2020.
- [27] Q. Wu and X. Chen, "Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring," *IEEE Transactions on Mobile Computing*, 2020.
- [28] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.
- [29] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.
- [30] Y. Ozturk and P. Jha, "A personalized home energy management system for residential demand response," in *4th International Conference on Power Engineering, Energy and Electrical Drives*, 2013.
- [31] T. Shoji and W. Hirohashi, "Personalized energy management systems for home appliances based on bayesian networks," *Journal of International Council on Electrical Engineering*, 2015.
- [32] J. Haj-Yahya, "Techniques for reducing the connected-standby energy consumption of mobile devices," in *2020 IEEE HPCA*, 2020.
- [33] G. P. Meyer, "An alternative probabilistic interpretation of the huber loss," in *Proceedings of the IEEE/CVF CVPR*, 2021.
- [34] P. A.-D.-V. Raj, M. Sudhakaran, P. P.-D.-A. Raj, *et al.*, "Estimation of standby power consumption for typical appliances," *Journal of Engineering Science and Technology Review*, vol. 2, no. 1, pp. 71–75, 2009.

- [35] “Texas electricity rates,” <https://comparepower.com>, 2020.
- [36] “Energy information administration,” <https://www.eia.gov/>, 2020.
- [37] W. Xu and H. Peng, “A hybrid modelling method for time series forecasting based on a linear regression model and deep learning,” *Applied Intelligence*, 2019.
- [38] L. Cao, “Support vector machines experts for time series forecasting,” *Neurocomputing*, vol. 51, pp. 321–339, 2003.
- [39] L. Wang, “Back propagation neural network with adaptive differential evolution algorithm for time series forecasting,” 2015.
- [40] I. Sülo, S. R. Keskin, G. Dogan, and T. Brown, “Energy efficient smart buildings: Lstm neural networks for time series prediction,” in *2019 International Conference on Deep-ML*, IEEE, 2019.
- [41] X. Xu, Y. Jia, Y. Xu, Z. Xu, S. Chai, and C. S. Lai, “A multi-agent reinforcement learning-based data-driven method for home energy management,” *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3201–3211, 2020.
- [42] A.-R. Al-Ali, I. A. Zualkernan, M. Rashid, R. Gupta, and M. Alikarar, “A smart home energy management system using iot and big data analytics approach,” *IEEE Transactions on Consumer Electronics*, 2017.
- [43] T. Ahmad, H. Chen, R. Huang, *et al.*, “Supervised based machine learning models for short, medium and long-term energy prediction in distinct building environment,” *Energy*, vol. 158, pp. 17–32, 2018.
- [44] W. Jiang and J. Luo, “Big data for traffic estimation and prediction: A survey of data and tools,” *Applied System Innovation*, vol. 5, no. 1, p. 23, 2022.
- [45] A. Sujith, G. S. Sajja, V. Mahalakshmi, S. Nuhmani, and B. Prasanalakshmi, “Systematic review of smart health monitoring using deep learning and artificial intelligence,” *Neuroscience Informatics*, vol. 2, no. 3, p. 100 028, 2022.
- [46] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, and J. L. Reyes Ortiz, “A public domain dataset for human activity recognition using smartphones,” in *Proceedings of the 21th international European symposium on artificial neural networks, computational intelligence and machine learning*, 2013, pp. 437–442.

- [47] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.
- [48] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, “Federated learning for internet of things: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, 2021.
- [49] A. F. Aji and K. Heafield, “Sparse communication for distributed gradient descent,” *arXiv preprint arXiv:1704.05021*, 2017.
- [50] P. P. Liang, T. Liu, L. Ziyin, *et al.*, “Think locally, act globally: Federated learning with local and global representations,” *arXiv preprint arXiv:2001.01523*, 2020.
- [51] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, “Protection against reconstruction and its applications in private federated learning,” *arXiv preprint arXiv:1812.00984*, 2018.
- [52] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, “Exploiting unintended feature leakage in collaborative learning,” in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019, pp. 691–706.
- [53] L. Lyu, H. Yu, X. Ma, *et al.*, “Privacy and robustness in federated learning: Attacks and defenses,” *arXiv preprint arXiv:2012.06337*, 2020.
- [54] L. Sun, J. Qian, and X. Chen, “Ldp-fl: Practical private aggregation in federated learning with local differential privacy,” *arXiv preprint arXiv:2007.15789*, 2020.
- [55] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” in *Annual international conference on the theory and applications of cryptographic techniques*, Springer, 2006, pp. 486–503.
- [56] T. Yu, E. Bagdasaryan, and V. Shmatikov, “Salvaging federated learning by local adaptation,” *arXiv preprint arXiv:2002.04758*, 2020.
- [57] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, “Semi-supervised knowledge transfer for deep learning from private training data,” *arXiv preprint arXiv:1610.05755*, 2016.

- [58] T. T. Nguyễn, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin, “Collecting and analyzing data from smart device users with local differential privacy,” *arXiv preprint arXiv:1606.05053*, 2016.
- [59] Y. Zhao, J. Zhao, M. Yang, *et al.*, “Local differential privacy-based federated learning for internet of things,” *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8836–8853, 2020.
- [60] L. Lyu, H. Yu, X. Ma, *et al.*, “Privacy and robustness in federated learning: Attacks and defenses, 2020,” *arXiv preprint arXiv:2012.06337*, 2012.
- [61] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, “Three approaches for personalization with applications to federated learning,” *arXiv preprint arXiv:2002.10619*, 2020.
- [62] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, “Federated evaluation of on-device personalization,” *arXiv preprint arXiv:1910.10252*, 2019.
- [63] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, “Federated multi-task learning,” *Advances in neural information processing systems*, vol. 30, 2017.
- [64] Y. Jiang, J. Konečný, K. Rush, and S. Kannan, “Improving federated learning personalization via model agnostic meta learning,” *arXiv preprint arXiv:1909.12488*, 2019.
- [65] V. Kulkarni, M. Kulkarni, and A. Pant, “Survey of personalization techniques for federated learning,” in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, IEEE, 2020, pp. 794–797.
- [66] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. 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*, 2021, pp. 54–66.
- [67] A. Li, J. Sun, X. Zeng, M. Zhang, H. Li, and Y. Chen, “Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking,” in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021, pp. 42–55.
- [68] J. Gao, M. Tang, T. Wang, and B. Campbell, “Pfed-ldp: A personalized federated local differential privacy framework for iot sensing data,” in *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, 2022, pp. 835–836.



- [69] J. Gao, W. Wang, F. Nikseresht, V. Govinda Rajan, and B. Campbell, “Pfdrl: Personalized federated deep reinforcement learning for residential energy management,” in *Proceedings of the 52nd International Conference on Parallel Processing*, 2023, pp. 402–411.
- [70] L. Tu, X. Ouyang, J. Zhou, Y. He, and G. Xing, “Feddl: Federated learning via dynamic layer sharing for human activity recognition,” in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021, pp. 15–28.
- [71] M. Kim, O. Günlü, and R. F. Schaefer, “Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 2650–2654.
- [72] R. Liu, Y. Cao, H. Chen, R. Guo, and M. Yoshikawa, “Flame: Differentially private federated learning in the shuffle model,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 8688–8696.
- [73] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [74] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [75] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.
- [76] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.
- [77] S. Niknam, H. S. Dhillon, and J. H. Reed, “Federated learning for wireless communications: Motivation, opportunities, and challenges,” *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, 2020.
- [78] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

- [79] K. Wei, J. Li, M. Ding, *et al.*, “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.
- [80] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani, “The architecture of complex weighted networks,” *Proceedings of the national academy of sciences*, vol. 101, no. 11, pp. 3747–3752, 2004.
- [81] M. Zhu and S. Gupta, “To prune, or not to prune: Exploring the efficacy of pruning for model compression,” *arXiv preprint arXiv:1710.01878*, 2017.
- [82] H. Zhou, J. Lan, R. Liu, and J. Yosinski, “Deconstructing lottery tickets: Zeros, signs, and the supermask,” *Advances in neural information processing systems*, vol. 32, 2019.
- [83] H. B. McMahan, G. Andrew, U. Erlingsson, *et al.*, “A general approach to adding differential privacy to iterative training procedures,” *arXiv preprint arXiv:1812.06210*, 2018.
- [84] M. Abadi, A. Chu, I. Goodfellow, *et al.*, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.
- [85] A. Bittau, Ú. Erlingsson, P. Maniatis, *et al.*, “Prochlo: Strong privacy for analytics in the crowd,” in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 441–459.
- [86] C. O’reilly, N. Gosselin, J. Carrier, and T. Nielsen, “Montreal archive of sleep studies: An open-access resource for instrument benchmarking and exploratory research,” *Journal of sleep research*, vol. 23, no. 6, pp. 628–635, 2014.
- [87] J. W. Lockhart, G. M. Weiss, J. C. Xue, S. T. Gallagher, A. B. Grosner, and T. T. Pulickal, “Design considerations for the wisdm smart phone-based sensor mining architecture,” in *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*, 2011, pp. 25–33.
- [88] S. Alam, T. Zhang, T. Feng, *et al.*, “Fedaiot: A federated learning benchmark for artificial intelligence of things,” *arXiv preprint arXiv:2310.00109*, 2023.
- [89] CIFAR-10, *The CIFAR-10 dataset*, <https://www.cs.toronto.edu/~kriz/cifar.html>.

- [90] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.
- [91] J. Sun, Y. Fang, and X. Zhu, "Privacy and emergency response in e-healthcare leveraging wireless body sensor networks," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 66–73, 2010.
- [92] T. Wu, M. Aldeer, T. Chowdhury, *et al.*, "The smart building privacy challenge," in *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2021, pp. 238–239.
- [93] J. Lola, C. Serrão, and J. Casal, "Towards transparent and secure iot: Improving the security and privacy through a user-centric rules-based system," *Electronics*, vol. 12, no. 12, p. 2589, 2023.
- [94] A. Aljerais, M. Barati, O. Rana, and C. Perera, "Privacy laws and privacy by design schemes for the internet of things: A developer's perspective," *ACM Computing Surveys (Csur)*, vol. 54, no. 5, pp. 1–38, 2021.
- [95] M. Gheisari, H. E. Najafabadi, J. A. Alzubi, *et al.*, "Obpp: An ontology-based framework for privacy-preserving in iot-based smart city," *Future Generation Computer Systems*, vol. 123, pp. 1–13, 2021.
- [96] M. A. Husnoo, A. Anwar, R. K. Chakraborty, R. Doss, and M. J. Ryan, "Differential privacy for iot-enabled critical infrastructure: A comprehensive survey," *IEEE Access*, vol. 9, pp. 153 276–153 304, 2021.
- [97] S. Ghayyur, Y. Chen, R. Yus, *et al.*, "Iot-detective: Analyzing iot data under differential privacy," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1725–1728.
- [98] C. Huang, S. Chen, Y. Zhang, W. Zhou, J. J. Rodrigues, and V. H. C. de Albuquerque, "A robust approach for privacy data protection: Iot security assurance using generative adversarial imitation learning," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 089–17 097, 2021.
- [99] W. Yao, H. Zhao, and H. Shi, "Privacy-preserving collaborative intrusion detection in edge of internet of things: A robust and efficient deep generative learning approach," *IEEE Internet of Things Journal*, 2023.

- [100] X. Yang and O. Ardakanian, "Privacy through diffusion: A white-listing approach to sensor data anonymization," in *Proceedings of the 5th Workshop on CPS&IoT Security and Privacy*, 2023, pp. 101–107.
- [101] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.
- [102] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [103] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in neural information processing systems*, vol. 32, 2019.
- [104] J. G. Wijmans and R. W. Baker, "The solution-diffusion model: A review," *Journal of membrane science*, vol. 107, no. 1-2, pp. 1–21, 1995.
- [105] N. Tamani and Y. Ghamri-Doudane, "Towards a user privacy preservation system for iot environments: A habit-based approach," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 2016, pp. 2425–2432.
- [106] J. Wang, Y. Chen, Y. Gu, Y. Xiao, and H. Pan, "Sensorygans: An effective generative adversarial framework for sensor-based human activity recognition," in *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018, pp. 1–8.
- [107] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *Advances in neural information processing systems*, vol. 32, 2019.
- [108] X. Li, J. Luo, and R. Younes, "Activitygan: Generative adversarial networks for data augmentation in sensor-based human activity recognition," in *Adjunct proceedings of the 2020 ACM international joint conference on pervasive and ubiquitous computing and proceedings of the 2020 ACM international symposium on wearable computers*, 2020, pp. 249–254.
- [109] N. Sivaroopan, D. Bandara, C. Madarasingha, G. Jourjon, A. P. Jayasumana, and K. Thilakarathna, "Netdiffus: Network traffic generation by diffusion models through time-series imaging," *Computer Networks*, p. 110 616, 2024.
- [110] G. Batzolis, J. Stanczuk, C.-B. Schönlieb, and C. Etmann, "Conditional image generation with score-based diffusion models," *arXiv preprint arXiv:2111.13606*, 2021.

- [111] L. Zhang, A. Rao, and M. Agrawala, “Adding conditional control to text-to-image diffusion models,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3836–3847.
- [112] R. Aloufi, H. Haddadi, and D. Boyle, “Privacy-preserving voice analysis via disentangled representations,” in *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, 2020, pp. 1–14.
- [113] S. Zhu, C. Zhang, and X. Zhang, “Automating visual privacy protection using a smart led,” in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 2017, pp. 329–342.
- [114] D. Chen, D. Irwin, P. Shenoy, and J. Albrecht, “Combined heat and privacy: Preventing occupancy detection from smart meters,” in *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, IEEE, 2014, pp. 208–215.
- [115] D. Feldman, C. Xiang, R. Zhu, and D. Rus, “Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks,” in *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, IEEE, 2017, pp. 3–16.
- [116] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, “Adversarial autoencoders,” *arXiv preprint arXiv:1511.05644*, 2015.
- [117] O. Hajihassnai, O. Ardakanian, and H. Khazaee, “Obscurenet: Learning attribute-invariant latent representation for anonymizing sensor data,” in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 40–52.
- [118] S. A. Osia, A. S. Shamsabadi, S. Sajadmanesh, *et al.*, “A hybrid deep learning architecture for privacy-preserving mobile analytics,” *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4505–4518, 2020.
- [119] M. Malekzadeh, R. G. Clegg, and H. 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)*, 2018, pp. 165–176.
- [120] A. Gupta, *Find out why drinking water while standing isn’t good for your health*, <https://www.healthshots.com/preventive-care/self-care/side-effects-of-drinking-water-while-standing/>, 2024.

- [121] V. Jakkula, “Tutorial on support vector machine (svm),” *School of EECS, Washington State University*, vol. 37, no. 2.5, p. 3, 2006.
- [122] M. Vrigkas, C. Nikou, and I. A. Kakadiaris, “A review of human activity recognition methods,” *Frontiers in Robotics and AI*, vol. 2, p. 28, 2015.
- [123] C. Jobanputra, J. Bavishi, and N. Doshi, “Human activity recognition: A survey,” *Procedia Computer Science*, vol. 155, pp. 698–703, 2019.
- [124] Z. Hussain, M. Sheng, and W. E. Zhang, “Different approaches for human activity recognition: A survey,” *arXiv preprint arXiv:1906.05074*, 2019.
- [125] A. Sunyaev and A. Sunyaev, “Cloud computing,” *Internet computing: Principles of distributed systems and emerging internet-based technologies*, pp. 195–236, 2020.
- [126] L. Qian, Z. Luo, Y. Du, and L. Guo, “Cloud computing: An overview,” in *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*, Springer, 2009, pp. 626–631.
- [127] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, “A review of applications in federated learning,” *Computers & Industrial Engineering*, vol. 149, p. 106 854, 2020.
- [128] Jonathan Ho, Ajay Jain, and Pieter Abbeel, “Denoising Diffusion Probabilistic Models,” in *NeurIPS*, 2020.
- [129] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, “Attention is All you Need,” in *NeurIPS*, 2017, pp. 5998–6008.
- [130] Y. Bai, A. Jones, K. Ndousse, *et al.*, “Training a helpful and harmless assistant with reinforcement learning from human feedback,” *arXiv preprint arXiv:2204.05862*, 2022.
- [131] S. Casper, X. Davies, C. Shi, *et al.*, “Open problems and fundamental limitations of reinforcement learning from human feedback,” *arXiv preprint arXiv:2307.15217*, 2023.
- [132] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, pp. 279–292, 1992.
- [133] Y. Hu, “Bsdgan: Balancing sensor data generative adversarial networks for human activity recognition,” in *2023 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2023, pp. 1–8.

- [134] Z. Yang, Y. Li, and G. Zhou, “Ts-gan: Time-series gan for sensor-based health data augmentation,” *ACM Transactions on Computing for Healthcare*, vol. 4, no. 2, pp. 1–21, 2023.
- [135] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.
- [136] A. Graves and A. Graves, “Long short-term memory,” *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.