Adaptive Differential Privacy for Decentralized Mobility Data Sharing and Forecasting

Fatima Zahra Errounda

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

November 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Ms. Fatima Zahra Errounda**

Entitled: **Adaptive Differential Privacy for Decentralized Mobility Data Sharing and Forecasting**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Gosta Grahne

_____ External Examiner
Dr. Benjamin C. M. Fung

_____ External to Program
Dr. Weiyi Shang

_____ Examiner
Dr. Anjali Agarwal

_____ Examiner
Dr. Hamou-Lhadj Wahab

_____ Supervisor
Dr. Yan Liu

Approved by _____
Dr. Yousef R. Shayan, Chair
Department of Electrical and Computer Engineering

_____ 2023

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Adaptive Differential Privacy for Decentralized Mobility Data Sharing and Forecasting

Fatima Zahra Errounda, Ph.D.

Concordia University, 2023

Mobility data is the cornerstone of crucial applications, including traffic monitoring, crowdsourcing, and social networks. However, research shows that publishing accurate mobility data aggregate may jeopardize the participants' privacy. As a robust and rigorous technique, differential privacy provides a quantifiable protection guarantee by injecting enough noise into the aggregates to make them resilient to privacy attacks while allowing learning and analysis.

The application of differential privacy raises two challenges that stem from mobility data characteristics. First, mobility data is usually spread across multiple organizations, whereas standard differential privacy relies on a centralized trusted curator. Secondly, mobility data is typically sequential, while the guarantee provided by differential privacy degrades with consecutive aggregating of the sensitive data. This thesis tackles these challenges for two application scenarios: decentralized mobility aggregate sharing and forecasting.

We leverage a distributed variant of differential privacy to enable decentralized mobility aggregate sharing where each organization obfuscates its dataset locally before sending it to the data curator. We use a sliding window approach to allocate the privacy budget to tackle the consecutive data access challenge. Moreover, we design an approximation strategy to calculate the closest private statistics to the current timestamp. We formally prove the privacy guarantee of our algorithms. Finally, we demonstrate that our solution enables decentralized statistical release with a robust privacy guarantee on two datasets.

Before addressing the privacy aspect of distributed mobility forecasting, we design a mobility vertical federated forecasting (MVFF) framework that allows the learning process to be jointly conducted over vertically partitioned data belonging to multiple organizations. Since each organization

only holds a location domain subset, none can tackle a forecasting model that covers the whole location domain. Moreover, distributed mobility data compromises the spatio-temporal correlation between locations hindering learning. Hence, reducing the forecasting accuracy. MVFF uses a local learning model for each organization to extract the embedded spatio-temporal correlation between its locations. A global model synchronizes with the local models to incorporate the correlation between all the organizations' locations. We investigate the performance of MVFF under four variations of local and global models. We compare the MVFF's performance to two other federated frameworks on real-life datasets: New York Bike and Yelp reviews, achieving better performances.

Finally, we design two adaptive differential privacy budget algorithms for each organization participating in collaborative mobility forecasting. We define a new metric to assess the different organizations' participation levels in the learning task and adjust the privacy budget accordingly. Then, we adapt each organization's privacy protection level (privacy budget) to the accuracy dynamics of the learning task. Lastly, we empirically evaluate our adaptive differential privacy budget algorithms using MVFF and two real-world datasets: a trajectory dataset collected in New York and Beijing over multiple months and a Yelp business review dataset.

Acknowledgments

I want to express my sincere gratitude to my supervisor, Dr. Yan Liu, who has inspired me constantly. Her management skills, vast technical knowledge, and stimulating discussions taught me a lot. I am also thankful for her guidance, kind words, and never giving up on me.

Contents

List of Figures x

List of Tables xv

1 Introduction 1

1.1 Problem Statement . . . 2

1.2 Summary of Methodology . . . 4

1.3 Thesis Contribution . . . 5

1.4 Publication List . . . 6

2 Background 7

2.1 Mobility Data . . . 7

2.1.1 Basic Definition . . . 7

2.1.2 Mobility Data Privacy . . . 8

2.2 Privacy Attacks against Mobility Data . . . 9

2.2.1 Attacks against Raw Mobility Data . . . 9

2.2.2 Attacks against Aggregate Mobility Data . . . 10

2.3 Differential Privacy . . . 11

2.3.1 Basic Notions . . . 11

2.3.2 Differential Privacy Properties . . . 12

2.3.3 Common Privacy Mechanisms . . . 13

2.3.4 Variations . . . 14

2.4 Deep Learning for Mobility Forecasting . . . 17
2.4.1 Mobility Forecasting . . . 17
2.4.2 Federated Learning . . . 21
2.5 Differential Privacy in Deep Learning . . . 21

3 Literature Review **24**
3.1 Differential Privacy for Mobility Data Collection . . . 25
3.2 Differentially Private Mobility Data Aggregation . . . 26
3.2.1 Singular Mobility Data Aggregate . . . 27
3.2.2 Sequential Mobility Data Aggregate . . . 28
3.3 Mobility Forecasting . . . 30
3.3.1 Centralized Mobility Forecasting Learning Models . . . 31
3.3.2 Federated Mobility Forecasting Learning Models . . . 32
3.3.3 Vertical Federated Learning . . . 33
3.4 Differentially Private Deep Learning . . . 34
3.4.1 Feature-Adaptive Learning Algorithms . . . 34
3.4.2 Budget Allocation . . . 35

4 Distributed Release of Mobility Aggregate with Differential Privacy **37**
4.1 Introduction . . . 37
4.2 Distributed Release of Mobility Aggregate with Differential Privacy . . . 40
4.2.1 Decision Step . . . 42
4.2.2 Perturbation Step . . . 44
4.2.3 Approximation Step . . . 45
4.3 Formal Privacy and Utility Analysis . . . 47
4.3.1 Privacy Analysis . . . 47
4.3.2 Utility Analysis . . . 51
4.4 Experimental Evaluation . . . 53
4.4.1 Datasets and Metrics . . . 53
4.4.2 Comparison Frameworks . . . 55

4.4.3 Experiment Design and Results . . . 57
4.4.4 Discussion . . . 62
4.4.5 Threat to Validity . . . 62
4.5 Summary . . . 62

5 A Mobility Forecasting Framework with Vertical Federated Learning 68
5.1 Introduction . . . 68
5.2 Problem Statement . . . 70
5.2.1 System Model . . . 71
5.2.2 Problem Formulation . . . 71
5.3 Mobility Vertical Federated Forecasting . . . 72
5.3.1 Overall Learning Framework . . . 72
5.3.2 Temporal Synchronization Module . . . 75
5.3.3 Spatial Synchronization Module . . . 76
5.4 Experimental Evaluation . . . 76
5.4.1 Datasets and Metrics . . . 77
5.4.2 Framework Implementation with Model Variations . . . 78
5.4.3 Comparison Frameworks . . . 79
5.4.4 Experiment design and results . . . 80
5.4.5 Discussion . . . 82
5.4.6 Threat to Validity . . . 83
5.5 Summary . . . 83

6 Adaptive Differential Privacy in Vertical Federated Learning for Mobility Forecasting 88
6.1 Introduction . . . 88
6.2 Privacy Threat Analysis . . . 91
6.2.1 Privacy Threat Taxonomy . . . 91
6.2.2 Threat Model . . . 93
6.2.3 Threat Analysis . . . 94
6.3 Adaptive Differential Privacy for Vertical Federated Learning Protocol Overview . 95

6.3.1 Overall Privacy Framework 95
6.3.2 Feature-Aware Privacy Budget Initialization 96
6.3.3 Adaptive Budgeting Scheme Algorithm 102
6.4 Case Study: Mobility Forecasting for Bike New York and Yelp Reviews 105
6.4.1 Features and Data Processing 106
6.4.2 Comparison Frameworks and Metrics 107
6.4.3 Experiment Design and Results 110
6.4.4 Discussion 114
6.4.5 Threat to Validity 115
6.5 Summary 115
6.6 Membership Inference Attack Results 116
6.6.1 Attack Details and Metrics 116

7 Conclusion and Future Directions **126**

Bibliography **129**

List of Figures

Figure 1.1 An application example of distributed location aggregate real-time monitoring and mobility forecasting. The monitoring system uses the private statistics captured and sent by each organization to control systems such as crowd monitoring and traffic. The learning system predicts human mobility based on the private location statistics captured by the organizations. 3

Figure 2.1 An overview of the mobility data flow from collection to publishing with the public. Location-tracking devices, such as mobile phones, cars, buses, and trains, share their location with a service provider. Aggregation and mining of these locations are crucial for understanding human mobility in various application domains, including transportation systems, social networks, and recommendation systems. . 8

Figure 2.2 Centralized setting vs. local setting of differential privacy: the privacy mechanism is applied by the curator in the centralized setting, whereas each user is responsible for applying the privacy mechanism in the local setting. 15

Figure 2.3 LSTM Cell structure: the gates are controlled by the input, the hidden state, and the hidden cell state at each time step. 18

Figure 2.4 An simple CNN architecture, comprised of just five layers O'Shea and Nash (2015). 20

Figure 2.5 Federated learning categorization: (a) horizontal data distribution results in local datasets that share the same attributes (b) vertical data distribution leads to datasets with distinct attributes but common sample IDs 23

Figure 4.1 The three steps of the collective local differential privacy algorithm: the similarity step compares the previous and current mobility aggregate, the approximation strategy returns the closest private release, and the perturbation step computes a privacy budget to calibrate the added noise to the accurate aggregate. 42

Figure 4.2 We illustrate the privacy budget allocation with an example, where w=3 and 6 timestamps. Assume that the decision step returns a dissimilarity for timestamps 1, 2, 3, 4, and 6. At timestamps 5, the statistics are similar. Therefore, the released statistics is approximated, and no budget is used for the perturbation step. For the first timestamp 1, the last privacy budget allocated is 0, and the forward sliding window projection has w timestamps. Consequently ${\epsilon_1}^f = \epsilon/6$. Also, the backward sliding window did not consume any privacy budget, then ${\epsilon_1}^b = \epsilon/4$. The minimum value of ${\epsilon_1}^b$ and ${\epsilon_1}^f$ is assigned to ${\epsilon_i}^p = \epsilon/6$. The same logic is used for timestamps 2, 3, and 4. We perform the same calculation for the budget distribution and the budget absorption approaches. For all the approaches, the privacy budget within a window of 3 timestamps never exceeds $\epsilon/2$. 46

Figure 4.3 Telecommunication dataset: Proximity detection devices capture the number of nearby users at each timestamp. To protect the participants' privacy, the collective differential privacy algorithm (Algorithm 1) is applied at each device. 54

Figure 4.4 Bikeshare dataset: The collective local differential privacy algorithm (Algorithm 1) is applied to the number of bike owners at each bike station. Then, the resulting private number of bike owners is shared. 54

Figure 4.5 Perturbation error vs ϵ 58

Figure 4.6 Effect of the underlying noise generation mechanism on the utility of the collective local differential algorithm 59

Figure 4.7 Perturbation error vs w 60

Figure 4.8 Effect of the approximation strategy on the collective local differential privacy algorithm utility 64

Figure 4.9 Effect of the window size w on the execution time of the collective local differential private algorithm (4.9a, 4.9b), the budget absorption approach (4.9c, 4.9d), and the budget distribution approach (4.9e, 4.9f) 65
Figure 4.10 Effect of the approximation strategy on the utility of budget distribution approach 66
Figure 4.11 Effect of the approximation strategy on the utility of the budget absorption approach 67
Figure 5.1 System model: our main focus is to design a learning framework that coordinates the learning between multiple organizations to make mobility forecasting. . 69
Figure 5.2 Building the training batch: a sequence of $\mathcal{I}$ mobility counts with consecutive timestamps is restructured into sequences with the historical data length $\mathcal{T}$. The targeted mobility count represents the ground truth prediction. 73
Figure 5.3 Illustration of the training procedure in the MVFF framework. The global model trains mobility data generated using the temporally synchronized local models' training parameters and the location domain map. The different colours on the location map indicate different location features (mobility count) coming from the corresponding organizations. 75
Figure 5.4 Mobility counts (ground truth) for Bike New York and Yelp review datasets. 85
Figure 5.5 Mobility forecast for Bike New York and Yelp review datasets. 86
Figure 5.6 Metrics vs temporal interval length for the Bike New York and Yelp review datasets 87
Figure 6.1 The privacy threat taxonomy details the characteristics of the attacker and target for privacy threat analysis. 92
Figure 6.2 The adopted threat model: from the standpoint of each organization, other organizations and the learning collaborators are outside the trust boundary and considered honest-but-curious adversaries. 93

Figure 6.3 Illustration of the AdaVFL protocol components and steps. Each organization exchanges information about its features with the learning collaborator through the feature-aware privacy initialization and feature contribution processing components before the start of the training process. The organizations then assign a total privacy budget for the entire training task. Then, it breaks it down into individual privacy budgets per training iteration using the adaptive budgeting scheme component. 96

Figure 6.4 Weighting strategies that assign an initial privacy budget to each organization based on the normalized feature impact values, which range from 0% to 100%. The plots illustrate the weighting strategies with their limit values. 101

Figure 6.5 The feedback control loop adjusts the privacy budget ρ_t at each training iteration t to the model's convergence dynamics through the feedback error E_t. The convergence of the model is assessed with the training loss. 103

Figure 6.6 Grid division into 16x8 regions for the New York bike dataset and 8x8 regions for the Yelp review datasets. Each region has timestamped mobility counts and corresponds to a single feature. 106

Figure 6.7 The privacy feature inference attack details: The attacker trains a model that takes the known feature values from compromised organizations and randomly generated features for the targeted organization as input. Then, the attacker model generates potential feature values. The loss is the comparison result between the targeted trained model's output given the guessed features and the true labels. . . . 108

Figure 6.8 The utility of the local model under fixed training epochs on Bike New York and Yelp review datasets. 119

Figure 6.9 The utility of the global model under fixed training epochs on Bike New York and Yelp review datasets. 120

Figure 6.10 The utility of the local model under a fixed privacy budget on Bike New York and Yelp review datasets. 121

Figure 6.11 The utility of the global model under a fixed privacy budget on Bike New York and Yelp review datasets. 122

Figure 6.12 The budget assigned at each iteration per solution captured for a fixed number of iterations. The beginning of the dotted lines indicates the iteration where the overall budget is exhausted. 123

Figure 6.13 Convergence results of with AdaVFL with different noise levels ($\epsilon = 1.4$, $\epsilon = 1.6$, $\epsilon = 1.8$) and without noise on Bike New York and Yelp reviews datasets. . 124

Figure 6.14 The effect of the algorithm parameters on utility. When mu is closer to 0, the budget allocation becomes less adaptive and behaves like the uniform budget scheme, leading to lower utility. When m is larger, it takes longer intervals to adapt the privacy budget, resulting in a slight decrease in utility. 125

Figure 6.15 The membership inference attack details: The learning collaborator receives the intermediate training parameters from the targeted organization and uses them as input for an unsupervised attack model. The model is composed of a convolutional neural network and several fully connected networks, an encoder, and a decoder. . . 125

List of Tables

Table 4.1 Used notations for distributed release of mobility aggregate with differential privacy 41
Table 5.1 Used notations for mobility forecasting with vertical federated learning . . . 71
Table 5.2 Basic analysis of the used datasets 78
Table 5.3 The metrics results on Bike New York dataset 81
Table 5.4 The metrics results on Yelp dataset 81
Table 6.1 The attack's success rate (MSE) comparison for the Bike New York dataset, the higher the value, the harder it is for the attacker to reconstruct the targeted features. (+%) represents a percentage increase in the attack's average metrics compared to the no-privacy solution. 111
Table 6.2 The attack's success rate (MSE) comparison for the Yelp reviews dataset, the higher the value, the harder it is for the attacker to reconstruct the targeted features. (+%) represents a percentage increase in the attack's average metrics compared to the no-privacy solution. 111
Table 6.3 A comparison of the accuracy of the learning collaborator attacker with various training epochs on Bike New York dataset. (↓%) represents the percentage decrease in the attack's accuracy compared to the no-privacy solution. 117
Table 6.4 A comparison of the accuracy of the learning collaborator attacker with various training epochs on Yelp review dataset. (↓%) represents the percentage decrease in the attack's accuracy compared to the no-privacy solution. 118

Table 6.5 A comparison of the accuracy of the learning collaborator attacker with various training data size on Bike New York dataset. (↓%) represents the percentage decrease in the attack's accuracy compared to the no-privacy solution. 118
Table 6.6 A comparison of the accuracy of the learning collaborator attacker with various training data size on Yelp review dataset. (↓%) represents the percentage decrease in the attack's accuracy compared to the no-privacy solution. 118

Chapter 1

Introduction

Understanding phenomena related to human mobility is critical for everyday life, from traffic monitoring (Ny, Touati, and Pappas (2014)) to urban planning (L. Liu, Biderman, and Ratti (2009)) and social networking (Sadeh et al. (2009)). With the emergence of positioning technologies, such as GPS, Wi-Fi, and Bluetooth, service providers can capture increasingly more accurate estimates of a user's location, leading to a more in-depth understanding of human mobility. However, this understanding comes with a privacy cost. Mobility data encapsulate the places users visit, live or work at and the people they meet, all of which may lead to unwanted information disclosure.

This thesis studies the problem of privacy-preserving analysis of mobility data that focuses on the following question: *How can we explore mobility data while preserving the privacy of individuals?* The privacy risks of publishing accurate individuals' location traces are undeniable, as shown in several studies (Thompson and Warzel (2022), Wernke, Skvortsov, Dürr, and Rothermel (2014)). One tempting assumption is that the privacy risks dissipate with data aggregation because the adversaries cannot distinguish each individual's records. However, an adversary may recover a single user's trajectory solely based on the number of users present at several locations at each time slot, as demonstrated in the privacy attack by F. Xu et al. (2017). This privacy breach is possible by exploiting the uniqueness and regularity of human mobility data statistics without prior knowledge. Moreover, sharing knowledge modelled based on private aggregate mobility data with sophisticated techniques, including deep learning, may seem innocuous. However, several privacy attacks show that an adversary can even reconstruct the sensitive data of any participant in the training data

based on the learned knowledge model (C. Chen, Lyu, Yu, and Chen (2022), Salem, Bhattacharyya, Backes, Fritz, and Zhang (2020)).

An essential step in tackling the privacy protection of individuals is determining a suitable privacy technique. Differential privacy (Dwork, Naor, Pitassi, and Rothblum (2010)) is the standard technique that enables acquiring useful aggregate information and knowledge about a group of people while revealing nothing about any single individual. For example, consider a dataset with information about a group of people and an adversary who knows the private data of every individual in the dataset except for one: Alice. The adversary is trying to learn about her private information by observing the aggregate or knowledge model constructed from the database. He may also have access to arbitrary side information about Alice. Differential privacy guarantees that regardless of the side information, the adversary will infer "almost" nothing new about Alice from the aggregate or knowledge model. This potential inference is measured by "epsilon," a critical parameter that allows the database curator to balance privacy loss and utility. We provide a more in-depth definition of differential privacy in Chapter 2.

Although differential privacy provides a robust guarantee to develop private algorithms for statistical analysis, and machine learning, it presents limitations when we wish to apply it to mobility data. Most work in differential privacy literature focuses on the centralized data curator setting, whereas mobility data is typically spread across many devices or organizations. Moreover, the "almost" nothing new that the adversary learns about an individual might become more significant when we consider the repeated aggregation of the same dataset. Indeed, the composition of any two differentially private algorithms remains private, with the privacy loss parameter, "epsilon," increasing gradually. This differential privacy property raises a challenge in the mobility data setting where time is an inherent characteristic. We review the state-of-the-art literature for differential privacy applications to mobility data in Chapter 3.

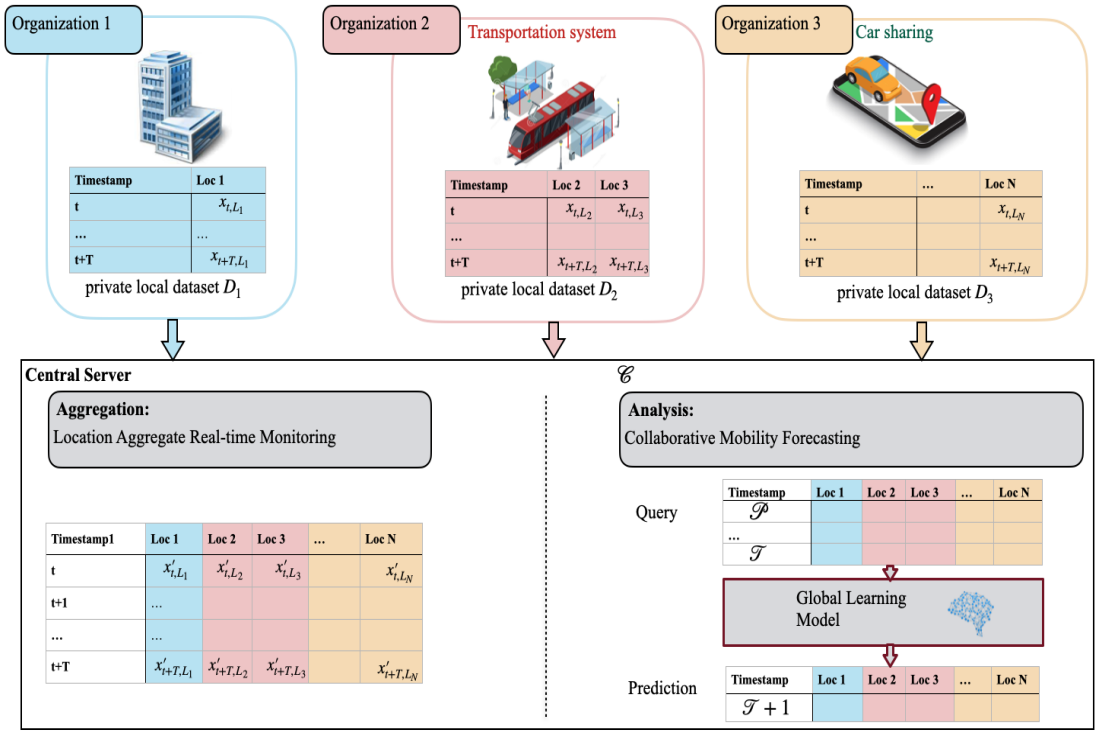


Figure 1.1: An application example of distributed location aggregate real-time monitoring and mobility forecasting. The monitoring system uses the private statistics captured and sent by each organization to control systems such as crowd monitoring and traffic. The learning system predicts human mobility based on the private location statistics captured by the organizations.

1.1 Problem Statement

Equipped with a better understanding of the challenges faced in applying differential privacy to mobility data, we can now define the problem that this thesis tackles. Let us consider the scenario illustrated in Figure 1.1. M organizations own mobility data that enables a wide range of practical applications, including crowd disaster prevention (Anzengruber, Pianini, Nieminen, and Ferscha (2013)), public transportation management (Rose (2006)), and even modelling epidemics' geographical spread (Oliver et al. (2020)). This mobility data may be the number of individuals in a given area with a time sampling rate. We denote $D_i = \{x_t\}_{t=1}^{T}$ as the organization, O_i, local dataset, where x_t represents the t-th row, and each column, x_{t,L_j}, corresponds to the number of users present at a location L_j at timestamp t.

A central server, $\mathcal{C}$, gathers this mobility data for real-time monitoring and learning. Each organization, O_i, aims at protecting its dataset, D_i, from privacy leaks. The trust boundary of O_i

excludes the central server and the other organizations. They are honest participants in the sharing or learning process who will not deviate from the defined protocol but will attempt to learn all possible information about the target organization's local dataset, D_i, from legitimately received messages. Hence, they are considered honest-but-curious adversaries (Paverd, Martin, and Brown (2014)).

With this threat model in mind, we can now identify the goals related to our scenarios:

- We must enable organizations to share mobility aggregate without compromising users' privacy and relying on a trusted data curator.
- The server should be able to collaborate with the M organizations to train a mobility forecasting model despite the vertical mobility data distribution.
- The forecasting training process must ensure the privacy of each organization's dataset.

This setting deviates from the standard differential privacy where the central server is trusted to curate the private datasets. Consequently, each organization is responsible for curating its private data before it leaves its trust boundary to be shared with the other organizations or the central server. However, the privacy expectation from different organizations may vary widely (G. Wang, Dang, and Zhou (2019)). Therefore, we need to answer the following research question: **How to enable differential privacy for heterogeneous organizations' privacy needs?**

Another challenge in applying differential privacy to this scenario stems from the sequential aspect of mobility data. Compared to a "one time" data release, this scenario requires protecting a user's privacy at each timestamp t instead of the entire course of the dataset. Moreover, each access to the dataset for knowledge modelling must be done in compliance with a privacy guarantee (quantified by the privacy loss parameter ϵ). Differential privacy provides a rigorous and provable quantification of privacy loss in the case of sequential applications. However, it is easy to see that the accumulation of privacy loss may lead to a poor privacy guarantee. Consequently, by treating the total privacy loss as a budget, a natural question arises: **How to allocate the privacy budget to prevent privacy leaks while maintaining utility?**

1.2 Summary of Methodology

Chapter 2 defines differential privacy's basics, tools for designing private algorithms, the basics of mobility forecasting and federated learning.

We review the state-of-the-art differential privacy solutions for mobility data to understand the research gaps better in Chapter 3. The fundamental elements of mobility data are location and time. Privacy concerns may arise during these elements' collection, aggregation, and analysis. We limit the review to the methods that allow differential privacy during consecutive aggregation and analysis based on neural networks for brevity. The remaining methods' analysis can be found in Errounda and Liu (2019).

We enable differential privacy for heterogeneous organizations' privacy needs in Chapter 4 and Chapter 6 for mobility aggregate sharing and forecasting, respectively. We leverage the local differential privacy variant for aggregate sharing and feature adaptive budget initialization for mobility prediction.

We set the stage for the mobility forecasting model in Chapter 5. We combine several neural network models with the vertical federated learning paradigm to enable organizations to train a global mobility prediction model.

We allocate the privacy budget to prevent privacy leaks while maintaining utility using a sliding window approach for mobility aggregate sharing, as detailed in Chapter 4. We leverage a feedback loop for mobility forecasting to adapt the privacy budget to the model's convergence state in Chapter 6. Chapter 7 concludes this thesis and highlights the future directions.

1.3 Thesis Contribution

The thesis goals are achieved through the following contributions:

- We address the heterogeneous organizations' privacy needs by leveraging local differential privacy, a distributed variant of differential privacy where each organization obfuscates its dataset locally before sending it to the data curator. We use a sliding window approach to allocate the privacy budget. Specifically, based on the similarity between two consecutive

aggregates, the algorithm skips the publication to avoid premature privacy budget exhaustion. Moreover, we design an approximation strategy to calculate the closest private aggregate to the current timestamp. We formally prove the privacy guarantee of our algorithms. Finally, we demonstrate in Errounda and Liu (2018) and Errounda and Liu (2021) that our solution enables decentralized statistical release with a robust privacy guarantee on two datasets (real-time counts of nearby users and historical counts of bike owners close to each bike station).

- We introduce the mobility vertical federated forecasting (MVFF) framework that enables knowledge modelling for organizations that hold only a subset of the location domain. Using a local learning model, each organization extracts the embedded spatio-temporal correlation between its locations. A global model synchronizes between the local models to incorporate the correlation between all the organizations' locations. We investigate the performance of MVFF under four different variations of local and global models. We verify in Errounda and Liu (2022) that the framework results in a performance gain compared to the state-of-the-art.

- We propose the Adaptive differential privacy for Vertical Federated Learning (AdaVFL) protocol to tackle the abovementioned questions. Second, we define a new metric to assess the different organizations' participation levels in the learning task and adjust the privacy budget to the organization's privacy needs. Then, we adapt each organization's privacy protection level (privacy budget) to the accuracy dynamics of the learning task. Finally, we evaluate our adaptive differential privacy budget algorithms over two real-world datasets in Errounda and Liu (2023): a trajectory dataset collected in New York and Beijing over multiple months and a Yelp business review dataset.

1.4 Publication List

Key publications of this thesis are:

- A detailed overview of differential privacy solution for mobility data in the 2019 IEEE International Conference on Big Data Security on Cloud Errounda and Liu (2019) (Chapter 3).

- A dynamic budget allocation for mobility aggregate sharing in the 2018 IEEE International Conference on Big Data Errounda and Liu (2018) (Chapter 4).
- An approximation strategy to improve the mobility aggregate sharing utility in the Future Generation Computer Systems journal (2021) Errounda and Liu (2021) (Chapter 4).
- A mobility forecasting framework with vertical federated learning in the 2022 IEEE Annual Computers, Software, and Applications Conference (COMPSAC) Errounda and Liu (2022) (Chapter 5).
- An adaptive differential privacy in vertical federated learning for mobility forecasting in the Future Generation Computer Systems journal (2023) Errounda and Liu (2023) (Chapter 6).

Chapter 2

Background

This chapter presents a definition of mobility data, then differential privacy's basics and tools for designing private algorithms. Finally, we introduce the basics of mobility forecasting and federated learning.

2.1 Mobility Data

Mobility data may be captured through many devices, including global positioning system tracking devices, radio-frequency identification devices, Bluetooth devices, and smartphones. As illustrated in Figure 2.1, this data undergoes several transformations through the collection, analysis, and sharing flow (X. Wu, Zhu, Wu, and Ding (2013)). We examine location data models that result from the diversity of tracking devices. Then, we introduce the different privacy threat models derived from the data flow.

2.1.1 Basic Definition

Mobility data describes the positioning of a device in space and time (B. Liu, Zhou, Zhu, Gao, and Xiang (2018)). The tuple $\prec$*identity; position; time*$\succ$, including the user's identity, the spatial information (position), and the temporal information (time), usually represents a user's position in space and time. Moreover, the location may have several presentations:

- Coordinates (a, b, c), where a represents latitude, b longitude, and c height, respectively. The

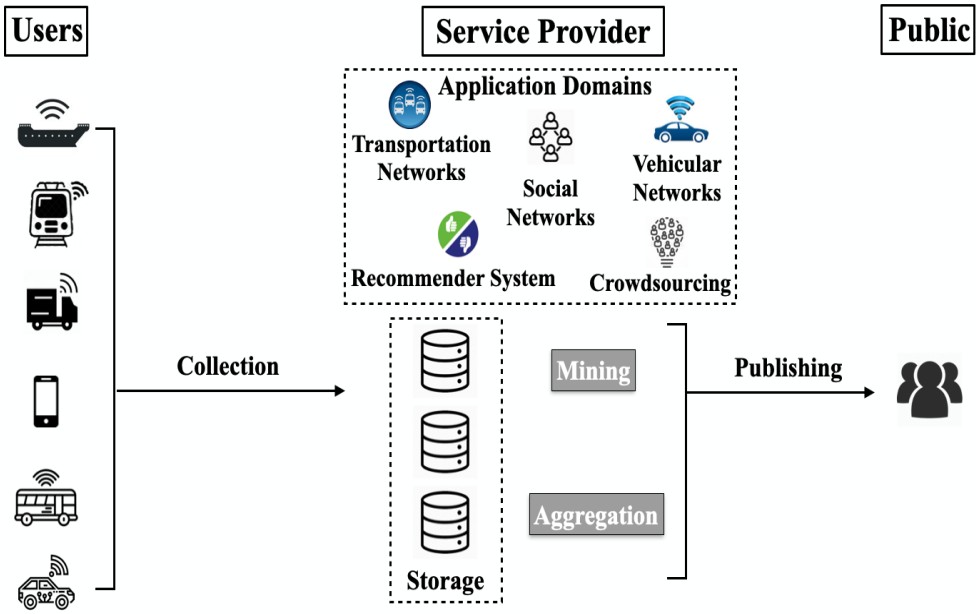


Figure 2.1: An overview of the mobility data flow from collection to publishing with the public. Location-tracking devices, such as mobile phones, cars, buses, and trains, share their location with a service provider. Aggregation and mining of these locations are crucial for understanding human mobility in various application domains, including transportation systems, social networks, and recommendation systems.

coordinate's values range between max and min with varying granularity.

- Identifier l_1 / Binary vector $(l_1, .., l_N)$, where N is a fixed number of predefined locations, l_1 to L_N. When the user is present in a location l_i, the identifier is l_i, and the binary vector holds 0 values in all indices except the i-th index.

2.1.2 Mobility Data Privacy

From the privacy perspective, we adopt Georgiadou, de By, and Kounadi (2019)'s location privacy definition:

> "Location privacy is the right to control the collection, access, recording, and usage of an individual's (location) information and determine when, how, and to what extent it is processed by others."

We derive the privacy threat models by combining the location privacy definition with the location data flow presented in Figure 2.1. A threat model (Xiong and Lagerström (2019)) provides a

structured way to analyze potential privacy threats by understanding an adversary's goal in attacking a system. We identify two threat models based on the access and usage of private data:

- Raw mobility data: The adversary tries to estimate the user's real locations using the locations reported by the mobile user.
- Mobility data aggregates: The adversary aims at reconstructing the users' real locations based on the aggregates shared by the service provider.

The adversary in both threat models can be honest-but-curious (Paverd et al. (2014)) who will not deviate from the defined exchange protocol but will attempt to learn all possible information from the legitimately received information. For instance, the adversary will not try to pose as a user to influence the reported locations or aggregates.

The first threat model considers the service provider and the public adversaries. They are outside the trust boundaries of the mobile user. A trust boundary (Stavroulakis and Stamp (2010)) describes any distinct boundary where all parties have equal trust. For example, the mobile user has a different level of trust than the service provider and the public in the first threat model. Therefore, it does not share the real locations with the service provider. The second threat model draws a trust boundary that includes the mobile users and the service provider. Consequently, the service provider computed the aggregates based on the real locations shared by the mobile users.

2.2 Privacy Attacks against Mobility Data

Several attacks (Gambs, Killijian, and del Prado Cortez (2010), Krumm (2007), Tockar (2014), Sadilek and Krumm (2012), Pyrgelis, Troncoso, and De Cristofaro (2017), and F. Xu et al. (2017)) uncover mobility data's vulnerability to privacy attacks. We present some basic privacy attacks against raw mobility data and mobility aggregates.

2.2.1 Attacks against Raw Mobility Data

Publishing accurate raw mobility data, such as GPS traces, involves serious privacy risks. For example, Gambs et al. (2010) designed an attack against mobility traces of taxi drivers in the San

Francisco Bay Area. The attack infers the points of interest of the taxi drivers by finding points where the taxi's GPS sensor was off for an extended time period (over 2 hours). Consequently, they could locate the driver's home and even confirm it by using a satellite view of the area showing the presence of a parked yellow cab.

Additionally, mobility traces may cause the inference of users' identities by associating an identity to each trace. For instance, Krumm (2007) used two months of mobility traces to infer users' home addresses with various heuristics. The attack used white pages to associate a person with a mobility trace. Moreover, Tockar (2014) showed how it was possible to stalk a celebrity using a taxi trips dataset and some background knowledge. The attack combines drop-off addresses of people frequently spending their night in a club and address information from Google and Facebook to pinpoint specific individuals with a high probability.

Finally, Sadilek and Krumm (2012) proposed a system to predict a user's location within one hour. The attack extracts repetitive patterns from mobility data using Fourier analysis. When the patterns were associated with a weekday and an hour in the day, the attacker reached a prediction accuracy between 77% to 93%.

2.2.2 Attacks against Aggregate Mobility Data

Sharing accurate aggregate mobility data with the public may enable trajectory extractions and membership inference privacy attacks. For example, Pyrgelis et al. (2017) studied the feasibility of membership inference attacks on aggregate location time series using a game-based definition of the adversarial task. They cast the game as a classification problem where a machine learning model distinguishes whether or not a target user is part of the aggregates. Naturally, the volume of contributed data, the regularity, and the particularity of users' mobility patterns play a crucial role in the attack's success.

Additionally, Zhang, Zhang, and Zhao (2020) presented an invasive attack that allows adversaries to launch membership inference attacks against aggregated location data without relying on any prior knowledge of the locations of victims. They train a binary classifier to infer whether a specific victim's location data is involved in the aggregation group based solely on the data aggregation's output.

Finally, F. Xu et al. (2017) demonstrated that individual user trajectories can be extracted from aggregate location statistics. In particular, the attack exploits the inherent characteristics of human mobility, namely, uniqueness and regularity. They model mobility patterns within different time periods (e.g., nighttime or daytime) and across different weekdays. Then, they combine these models to extract user trajectories using only public characteristics of the dataset as prior knowledge.

2.3 Differential Privacy

Differential privacy (Dwork, McSherry, Nissim, and Smith (2006)) is a formal notion requiring that any individual in the dataset has little influence on the information output by the algorithm. Specifically, a randomized algorithm compliant with differential privacy guarantees that the change to any single data record in the dataset does not change the probability of any event (based on the output) by much.

In this section, we first use the standard differential privacy setting, where a trusted curator with access to a sensitive dataset is assumed, to introduce the basic notions. Then, we highlight some of the most relevant properties of differential privacy. Next, we present the basic mechanisms to achieve differential privacy. Finally, we introduce some differential privacy variations that enable its distributed application and utility improvement.

2.3.1 Basic Notions

Let us consider a dataset, $\mathcal{D}$, and a query, q, as a function of the dataset. A data analyst asks the data curator a set of queries, and the data curator provides the answers by running the privacy mechanism, $\mathcal{M}$. To formally introduce differential privacy, we must first define the neighbouring relationship among databases. Formally:

Definition 1 (Neighbourhood)**.** *(Dwork et al. (2006)) Two datasets $\mathcal{D}$ and $\mathcal{D}'$ are neighbours if they differ in at most one element.*

Differential privacy requires the privacy mechanism, $\mathcal{M}$, to have "close" output distributions on any pair of neighbouring datasets as follows:

Definition 2 (Differential Privacy (Dwork et al. (2006))). *$\mathcal{M}$ satisfies (ϵ,δ)-differential privacy if for all neighbouring datasets $\mathcal{D}$ and $\mathcal{D}'$ and all $\mathcal{O} \in Range(\mathcal{M})$ the following holds:*

$$Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{O}] \leq e^{\epsilon} \times Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{O}] \ + \delta \tag{1}$$

The parameter ϵ controls the privacy loss incurred by observing the mechanism's output. Specifically, when $\delta = 0$, for an output o, ϵ is the upper bound of the following quantity:

$$ln(\frac{Pr[\mathcal{M}(\mathcal{D}) = o]}{Pr[\mathcal{M}(\mathcal{D}') = o]}) \tag{2}$$

Therefore, when ϵ goes to 0, it becomes harder for an adversary to distinguish if o originated from $\mathcal{D}$ or $\mathcal{D}'$. Let us consider an adversary that knows the entire dataset except for Alice's data and knows it is either in $\mathcal{D}$ or $\mathcal{D}'$. Differential privacy stipulates that he learns "almost" nothing new about Alice by observing the output, hence, protecting her privacy. This indistinguishability guarantees that the privacy of any single individual is protected even against an adversary equipped with side information. Consequently, the smaller the value of ϵ is, the better privacy guarantee $\mathcal{M}$ provides. If $\delta = 0$, we say that $\mathcal{M}$ is ϵ-differentially private.

When δ is non-zero, it indicates the probability that any individual suppers privacy loss exceeding ϵ. Indeed, (ϵ,δ)-differential privacy guarantees that, with probability at most 1-δ, the privacy loss does not exceed ϵ.

2.3.2 Differential Privacy Properties

We emphasize two of the most impactful properties of differential privacy. First, differential privacy is immune to post-processing, meaning that an adversary cannot increase his knowledge about an individual's private data solely based on the privacy mechanism output. For example, an adversary cannot combine a data-independent mapping f with an ϵ-differentially private mechanism $\mathcal{M}$ to gain more knowledge. In other words, $f \circ \mathcal{M}$ is also an ϵ-differentially private mechanism.

Next, we discuss the compositional property of differential privacy. Let us consider several (ϵ,δ)-differentially private mechanisms, and we wish to evaluate the privacy loss resulting from

composing these mechanisms. The composition theorem demonstrates that the privacy losses add up as follows:

Theorem 1 (Basic Composition (Dwork, Roth, et al. (2014))). *Consider mechanism $\mathcal{M}$ that provides (ϵ,δ)-differential privacy. A k sequence of $\mathcal{M}$ over a dataset $\mathcal{D}$ provides ($k\epsilon$,$k\delta$)-differential privacy.*

2.3.3 Common Privacy Mechanisms

We present three of the basic differentially private mechanisms. First, the Laplace mechanism answers real-valued queries with ϵ-differential privacy. The mechanism relies on the Laplace Distribution centred at 0 with scale b as the distribution with probability density function:

$$Lap(z|b) = \frac{1}{2b} exp(-\frac{|z|}{b}) \tag{3}$$

The Laplace mechanism is described as follows:

Definition 3 (Laplace Mechanism (Dwork et al. (2006))). *For a query q, a mechanism $\mathcal{M}(x) = q(x) + Lap(\frac{\Delta_1 q}{\epsilon})$ satisfies ϵ-differential privacy.*

Where Δ_1 represents the ℓ_1 sensitivity, it captures the magnitude by which a single individual's data can change the function q as follows:

$$\Delta_1 q = \max_{\mathcal{D},\mathcal{D}'} \parallel q(\mathcal{D}) - q(\mathcal{D}') \parallel_1 \tag{4}$$

for all neighbouring $\mathcal{D}$ and $\mathcal{D}'$.

Since the query, q, may output non-real-valued values, such as strings or trees, adding numerical noise is no longer sensible. The exponential mechanism solves this problem by defining a score function $u : \mathcal{R} \to \mathbb{R}$ that assigns a real-valued score to each possible query output. Then, it samples an output with a probability proportional to the sensitivity of the score function. Formally:

Definition 4 (Exponential Mechanism (McSherry and Talwar (2007))). *For a given score function $u : \mathcal{R} \to \mathbb{R}$, the mechanism $\mathcal{M}(\mathcal{D}, u, \mathcal{R})$ that selects and outputs an element $r \in \mathcal{R}$ with a probability proportional to $exp(\frac{\epsilon u(\mathcal{D},r)}{2\Delta})$ satisfies ϵ-differential privacy.*

where Δ is the sensitivity of u, defined as:

$$\Delta = \max_{\mathcal{D},\mathcal{D}':\|\mathcal{D}\triangle\mathcal{D}'\|=1,u:\mathcal{R}} \| u(\mathcal{D},r) - u(\mathcal{D}',r) \| \tag{5}$$

Finally, the Gaussian mechanism releases private perturbations of queries using the normal distribution as follows:

Definition 5 (Gaussian Mechanism (Dwork et al. (2014))). *For a query q, $\epsilon, \delta \in (0,1)$, and c a number such that $c^2 \geq 2log(1.25/\delta)$, the Gaussian mechanism $\mathcal{M}(x) = q(x) + \mathcal{N}(0,\sigma)$ with $\sigma \geq c\Delta_2(q)/\epsilon$ satisfies (ϵ,δ)-differential privacy.*

where Δ_2 is the ℓ_2 sensitivity, defined as:

$$\Delta_2 q = \max_{\mathcal{D},\mathcal{D}'} \| q(\mathcal{D}) - q(\mathcal{D}') \|_2 \tag{6}$$

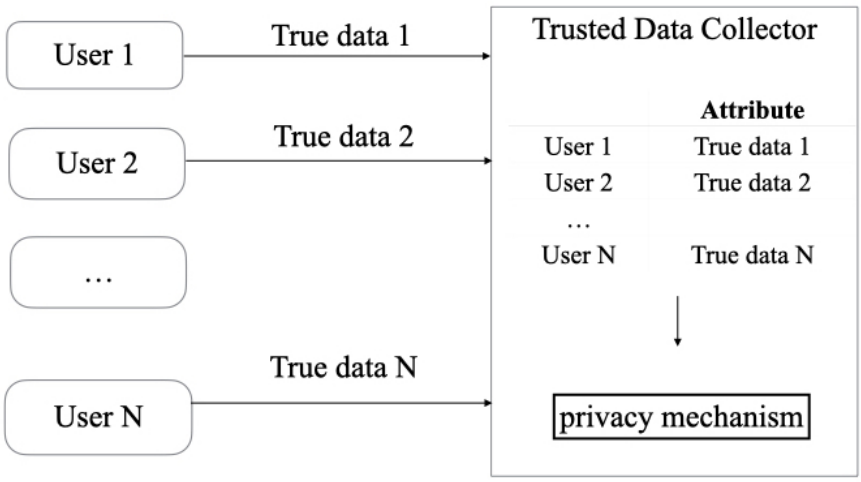
2.3.4 Variations

We will highlight two variations of differential privacy that we leverage in answering the proposed research questions.

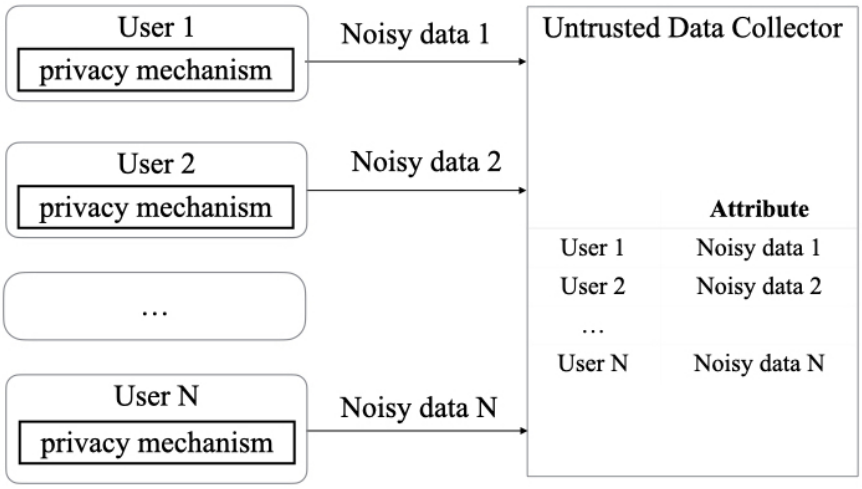
Local Differential Privacy

Unlike the centralized setting, local differential privacy does not involve a central trusted data curator. Instead, each user adds noise to his original data before sending it to the data collector for statistical analysis. Holding a central curator can become an adversary target in the case of privacy breaches. Therefore, this model is attractive for distributed implementations.

The randomized response is a privacy mechanism widely adopted for local differential privacy (Erlingsson, Pihur, and Korolova (2014) and Ding, Kulkarni, and Yekhanin (2017)). It was initially used in Warner (1965) to survey people's "yes or no" opinions about a sensitive question. Because the user gives a randomized response, the surveyors cannot determine the individual's true answer but can still extract valuable statistics (Dwork et al. (2014)). The mechanism is formally defined below:



(a) Centralized differential privacy setup



(b) Local differential privacy setup

Figure 2.2: Centralized setting vs. local setting of differential privacy: the privacy mechanism is applied by the curator in the centralized setting, whereas each user is responsible for applying the privacy mechanism in the local setting.

Definition 6 (Randomized Response)**.** *The randomized response mechanism $\mathcal{M}$ uses a biased coin to answer the query, q, truthfully if the coin turns head (with a probability of p). Otherwise, the mechanism answers with the opposite of the true answer (with a probability of $1-p$). Therefore, the randomized response satisfies ϵ-local differential privacy (ϵ-LDP) with the following value of p:*

$$p = \frac{e^{\epsilon}}{1+e^{\epsilon}} \tag{7}$$

Zero-Concentrated Differential Privacy

Zero-concentrated differential privacy (zCDP) is introduced by Bun and Steinke (Bun and Steinke (2016)) to solve the problem of mechanism composition with differing (ϵ_i,δ_i)-differential privacy guarantees. Indeed, the advanced composition theorem (Dwork, Rothblum, and Vadhan (2010)) demonstrates that the privacy loss in a composition of k ϵ-differentially private mechanisms grows roughly as $\sqrt{k}\epsilon$, instead of $k\epsilon$. This improvement allows us to achieve the same privacy guarantee for more frequent data aggregation. However, computing the tightest possible privacy guarantee for differing (ϵ_i,δ_i)-differentially private mechanisms is #P-hard (Murtagh and Vadhan (2015)). Therefore, zero-concentrated differential privacy is a variation that shifts the focus onto the privacy loss random variable for a privacy mechanism, $\mathcal{M}$, defined below:

$$L^{(o)} = log\frac{Pr[\mathcal{M}(x) = o]}{Pr[\mathcal{M}(x') = o]} \tag{8}$$

for an output $o \in range(\mathcal{M})$

zCDP uses the Rnyi divergence between probability distributions to capture the requirement that the privacy loss random variable is subgaussian. Rnyi divergence can be thought of as a measure of dissimilarity between distributions. This requirement enables zCDP to provide a more accurate analysis of the cumulative loss for multiple computations. Below is the formal definition of zCDP:

Definition 7 (Zero-Concentrated Differential Privacy (zCDP) (Bun and Steinke (2016)))**.** *A randomized mechanism $\mathcal{M}$ is ρ-zero concentrated differentially private if for all $\alpha \in (1, \infty)$ and any two neighbouring datasets $\mathcal{D}$ and $\mathcal{D}'$:*

$$D_\alpha(\mathcal{M}(\mathcal{D})||\mathcal{M}(\mathcal{D}')) \triangleq \frac{1}{\alpha - 1} log(\mathbb{E}[e^{(\alpha-1)L^{(o)}}]) \leq \alpha\rho \tag{9}$$

where $D_\alpha(\mathcal{M}(\mathcal{D})||\mathcal{M}(\mathcal{D}'))$ is α-Rnyi divergence between the distributions of $\mathcal{M}(\mathcal{D})$ and $\mathcal{M}(\mathcal{D}')$, and $L^{(o)}$ is the privacy loss random variable

It is important to note that if a mechanism satisfies ρ-zCDP, then it also satisfies (ϵ, δ)-differential privacy, for any $\delta > 0$, where:

$$\epsilon = \rho + 2\sqrt{\rho\ log(1/\delta)} \tag{10}$$

Moreover, the Gaussian mechanism introduced in Definition 5 satisfies ρ-zCDP, where:

$$\rho = \frac{1}{2\sigma^2} \tag{11}$$

Furthermore, ρ-zCDP also satisfies the composition theorem as follows:

Theorem 2 (Sequential Composition Bun and Steinke (2016)). *Mechanisms $\mathcal{M}_i$ guarantee ρ_i-zCDP, respectively. The composition of $\mathcal{M}_i$ in a sequence over a dataset $\mathcal{D}$ guarantees $\Sigma\rho_i$-zCDP.*

2.4 Deep Learning for Mobility Forecasting

Deep neural networks are remarkably effective for many machine learning tasks, such as producing a prediction in response to a query (Jordan and Mitchell (2015)). They rely on parameterized functions from inputs to outputs as compositions of multiple layers of basic building blocks, including affine transformations and simple non-linear functions. By varying the parameters of these blocks, we can "train" such a parameterized function to fit any given finite set of input/output examples. The training data may take the form of a collection of features and labels (Jordan and Mitchell (2015)). The inputs in the training data may be classical vectors or more complex objects, such as documents (Khan, Baharudin, Lee, and Khan (2010)), images (Krizhevsky, Sutskever, and Hinton (2012)), DNA sequences (Sahakyan et al. (2017)), or graphs (Scarselli, Gori, Tsoi, Hagenbuchner, and Monfardini (2008)).

Deep learning typically defines a loss function F representing the penalty for mismatching the training data. The loss $F(\theta)$ on parameters θ represents the average of the loss $F(\theta) = \frac{1}{N}\sum_{i=1}^{N} F(\theta, x_i)$ over the training samples $\{x_1, x_2, ..., x_N\}$. The training process aims at finding θ with an acceptably small loss. In practice, complex networks' loss function is usually non-convex and difficult to minimize. Therefore, the minimization is often done by the mini-batch stochastic gradient descent (SGD) algorithm as follows: at each step, i, a batch B_i of random samples is formed, and the gradient $\nabla_\theta F(\theta)$ is estimated to $g_i = \frac{1}{|B_i|}\sum_{x\in B_i} \nabla_\theta L(\theta, x)$. Finally, θ is updated towards a local minimum following the gradient direction $-g_i$.

This section focuses on the neural networks usually used for mobility forecasting and the federated learning that caters to the decentralized paradigm.

2.4.1 Mobility Forecasting

Due to the complex spatiotemporal relations in mobility data, combinations of several learning models are used in the state-of-the-art to achieve a precise prediction. Below are their definitions.

Long-Short Term Memory

Long-Short Term Memory (LSTM) is a class of recurrent neural networks, initially introduced by Hochreiter and Schmidhuber (Hochreiter and Schmidhuber (1997)) to correlate values from earlier stages for future use. The model's parameters are shared at different time steps, making it flexible in terms of sequence lengths. Given an input sequence represented by $(x_1, ..., x_T)$, the output sequence y_t is obtained in a basic LSTM network by iteratively computing the following equations for $t = 1, ..., T$:

$$h_t = LSTM(h_{t-1}, x_t; \theta)$$
$$y_t = \theta h_t + b_y$$

where the θ denotes the different weight matrices, b_y, the bias vector for the output y_t, and, h, the hidden state. This learning model propagates or forgets information over a long and recurrent training period to improve the prediction performance, as depicted in Figure 2.3. At each time step, the network's output is influenced by both the input and memory at the current time step. Then, the next time step's memory is updated by processing the current output through a non-linear activation function. LSTM is widely used in mobility forecasting, such as crowd prediction (Singh, Determe, Horlin, and De Doncker (2020), W. Li et al. (2019)), and traffic flow analysis (Y. Liu, James, Kang, Niyato, and Zhang (2020)).

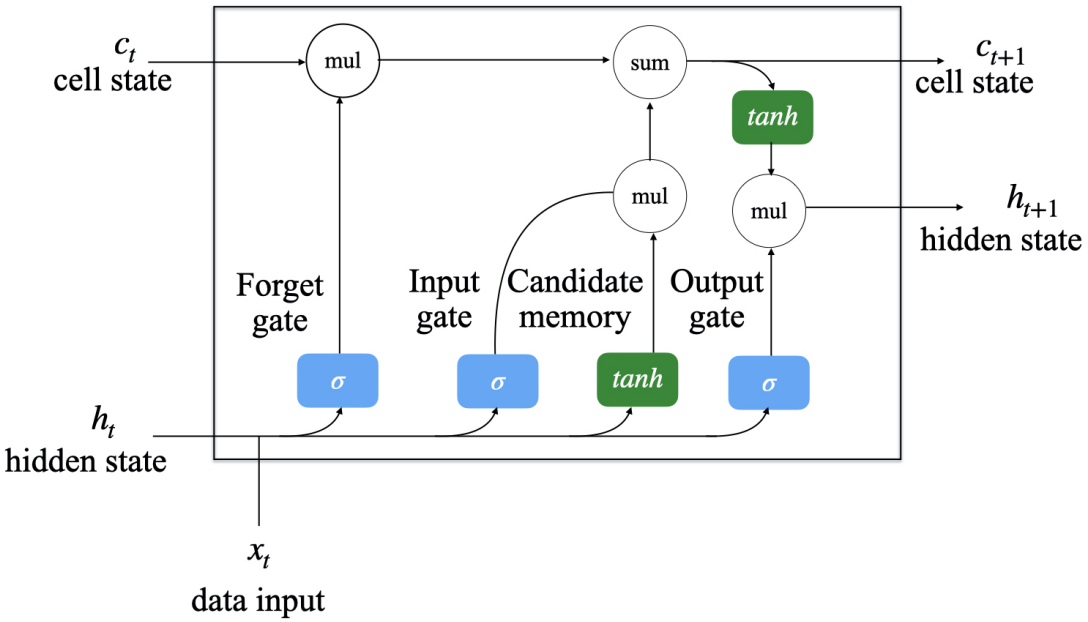


Figure 2.3: LSTM Cell structure: the gates are controlled by the input, the hidden state, and the hidden cell state at each time step.

Gated Recurrent Unit

Gated Recurrent Unit (GRU) is another recurrent neural network variant that handles time-series data. It was proposed by Cho et al. (K. Cho, Van Merriënboer, Bahdanau, and Bengio (2014)) in 2014. Similar to LSTM, GRU cells have a mechanism to judge whether a piece of information is valuable or not. However, GRU has less external gating signal in the interpolation. A typical structure of GRU cell has two data gates to control the data: the reset gate and the update gate. The reset gate performs the task of forgetting useless information and remembering the new important information. The update gate determines what portion of the previous state will be used as input. This optimization saves one gating signal and the associated parameters. GRUs are widely used in understanding human mobility, such as next location prediction (Lin, Liu, Li, and Zuo (2021)), crowd prediction (Zhou et al. (2020)), and traffic anomaly detection (Tao, Peng, Zhao, Wang, and Liu (2020)).

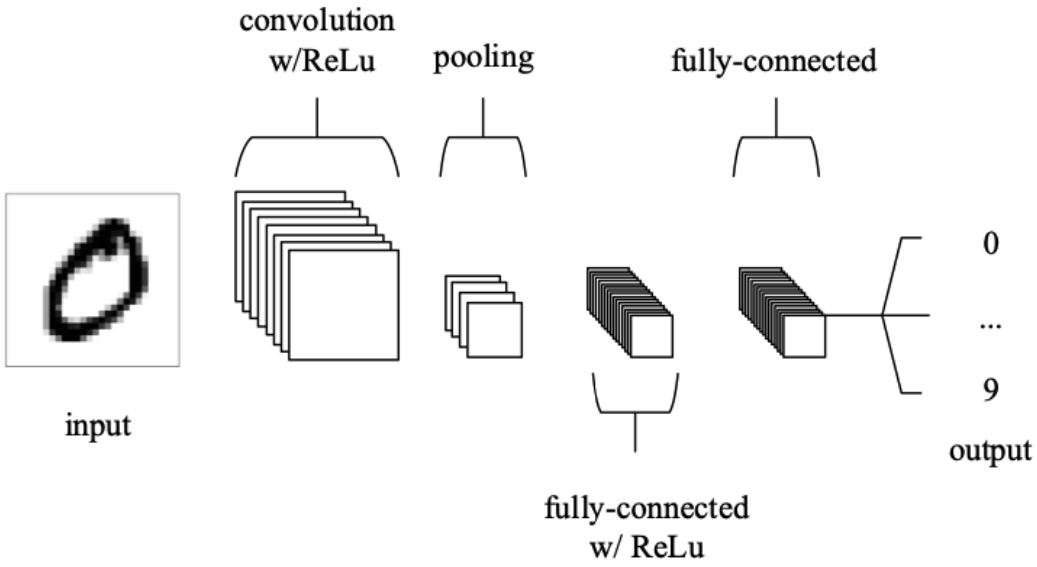


Figure 2.4: An simple CNN architecture, comprised of just five layers O'Shea and Nash (2015).

Convolutional Neural Networks

Convolutional Neural Network (CNN) (Krizhevsky et al. (2012)) is a class of neural networks that mimics the connectivity pattern between neurons in the human visual cortex. CNNs extract localized spatial features and compose them to create highly expressive representations. CNNs comprise three types of layers: convolutional, pooling, and fully-connected layers. These layers are stacked to form a CNN architecture. A simplified CNN architecture for MNIST classification is illustrated in Figure 2.4. The convolutional layer uses the scalar product to determine the output of neurons connected to local regions of the input. The pooling layer performs downsampling along the spatial dimensionality of the given input to reduce the number of parameters within that activation. Finally, the fully-connected layers produce class scores or predictions from the activations.

CNNs are widely applied in the field of computer vision and image recognition fields (Lawrence, Giles, Tsoi, and Back (1997), Qin et al. (2018)). CNNs are generally adopted in mobility prediction by converting human movement flow to images describing time and space relations. CNNs are used in a variety of mobility domains, such as indoor localization (X. Song et al. (2019)), intelligent transportation systems (G. Liu, Yin, Jia, and Xie (2017)), and location-based marketing (B. Zhu, Tang, Mao, and Yang (2020)).

Graph Neural Network

Graph Neural Network (GNN) (Scarselli et al. (2008)) is a category of neural networks designed to process data represented by a graph data structure. A graph is any set of objects and the connections between them. For example, images can be expressed as graphs where each pixel represents a node and is connected via an edge to adjacent pixels. An adjacency matrix is a way of visualizing the connectivity of a graph. Similarly, mobility data can be represented by an undirected graph where each region is a node, and the edges connect neighbouring regions. Graph attention networks (Veličković et al. (2018)) are a variant of GNNs that learn to weight the information from different neighbours with attention scores learned by a multi-head self-attention mechanism.

Unlike CNN, GNN can operate on graphs of arbitrary size and complex topology. GNNs are used in several mobility domains, such as in capturing location transition patterns in location trajectory (Sun et al. (2021)), modelling the spatiotemporal nature of the spread of infectious diseases (Roy et al. (2021)), and predicting traffic flow (X. Wang et al. (2020)).

2.4.2 Federated Learning

Federated learning (Konečnỳ, McMahan, Ramage, and Richtárik (2016)) is a nascent machine learning paradigm where many organizations jointly train a model under the orchestration of a central server while keeping the training data decentralized. An organization in federated learning computes an update of its local model based on its dataset and sends the updates to the server (Konečnỳ et al. (2016)). The server then computes a new global model based on the organizations' model updates (Konečnỳ et al. (2016)). Federated learning can be classified into horizontal and vertically federated learning (Q. Yang, Liu, Chen, and Tong (2019)) depending on the distribution of features and labels, as illustrated in Figure 2.5.

In horizontal federated learning, organizations share identical features and label space but differ in the samples' space (Q. Yang et al. (2019)). For instance, a mobile service jointly trains a next-word prediction model using mobile phone users' data (Hard et al. (2018)). Hence, mobile users' models share the same textual features; however, their locally stored text differs.

Conversely, organizations in vertical federated learning typically only have a subset of the

feature space that may overlap (Q. Yang et al. (2019)). For example, consider a bank and e-commerce that aim to build a prediction model for product purchases based on user and buying history (T. Chen, Jin, Sun, and Yin (2020)). However, the bank holds the user's earnings and spending behaviour, while the e-commerce company retains the user's purchasing history. Consequently, their feature spaces are very different despite the shared prediction model.

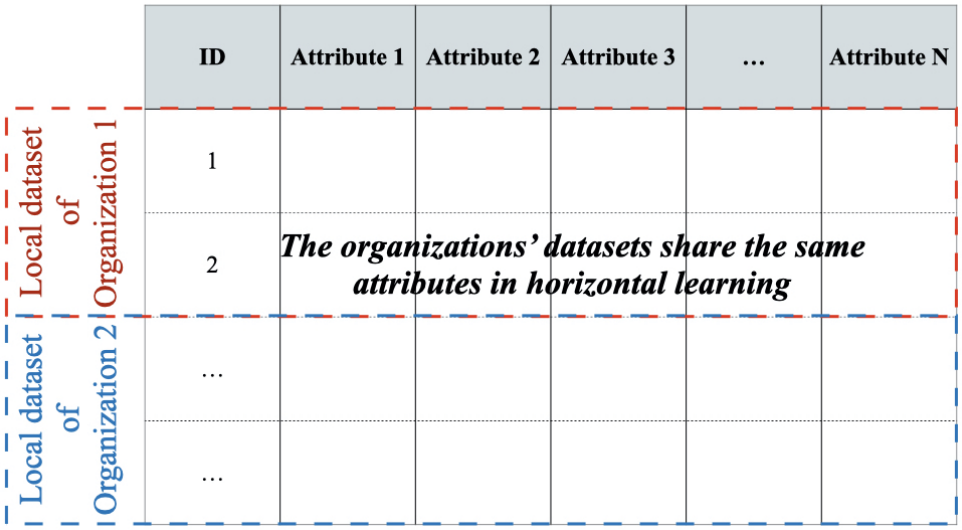
2.5 Differential Privacy in Deep Learning

One can apply differential privacy mechanisms within the deep learning framework: input perturbation, output perturbation, objective perturbation, and optimization algorithm alteration. In input perturbation, the result of the training dataset perturbation (using Laplace or Gaussian mechanism) is used to train the deep learning model (Duchi, Jordan, and Wainwright (2013)). In output perturbation techniques, the learning model is first trained, then perturbed using the exponential mechanism (Bassily, Smith, and Thakurta (2014)) or Laplace or Gaussian noise (X. Wu et al. (2017)). In objective perturbation techniques, the model is trained over the perturbed objective function with Laplace or Gaussian noise (Zhang, Zhang, Xiao, Yang, and Winslett (2012)). The fourth method involves adding noise to the model's gradient at each stochastic gradient descent (SGD) iteration (Abadi et al. (2016), S. Song, Chaudhuri, and Sarwate (2013)). We discuss the fourth method in this section.

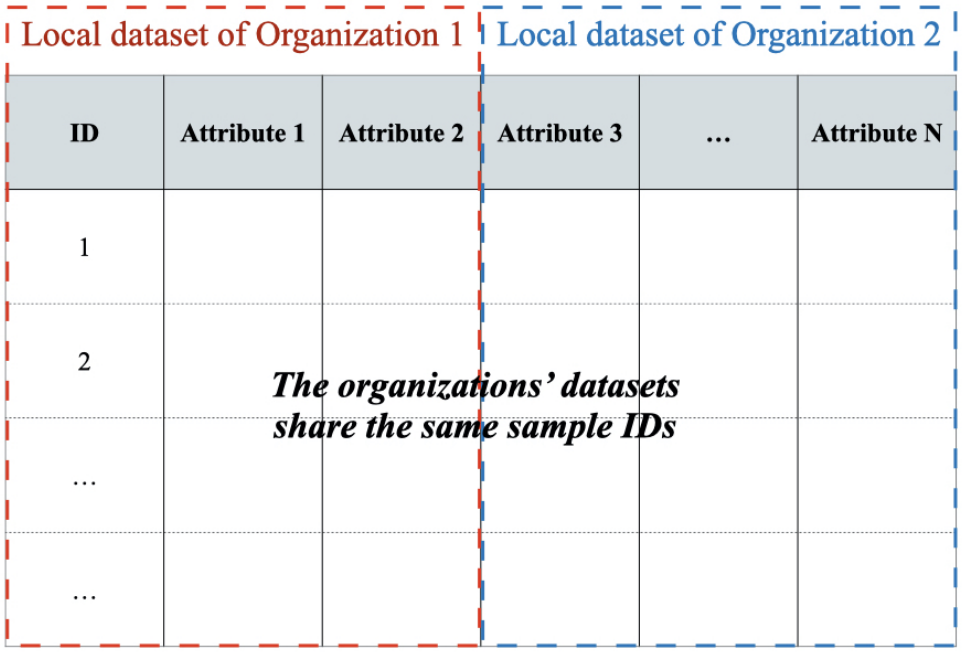
Gradient perturbation obfuscates the influence of an input sample on the final model. Because of the complexity of deep neural networks, it is hard to compute the gradient's sensitivity to calibrate the added privacy noise. Therefore, gradients are processed by clipping the $L2$-norm to a desired value C. The role of the clipping parameter is to limit the influence that a single sample can have on the computed gradients and, consequently, on the model. The added noise is generated with a Gaussian mechanism proportional to the clipping value. Because each SGD step is differentially private, the final model is also differentially private based on the composition property of differential privacy (Theorem 1).

In the federated setting, maintaining the private data locally for each organization does not entirely assure their privacy since several adversarial attacks can still compromise them (Bhagoji,

Chakraborty, Mittal, and Calo (2019)). The training process in federated learning requires that each organization trains the local model using its private data and adds carefully calculated noise to its model's output. This output is then sent to the untrusted server and fed to the server's learning model. Based on the post-processing property (defined in subsection 2.3.2) of differential privacy, the resulting server's learning model preserves privacy.



(a) Horizontal Federated Learning



(b) Vertical Federated Learning

Figure 2.5: Federated learning categorization: (a) horizontal data distribution results in local datasets that share the same attributes (b) vertical data distribution leads to datasets with distinct attributes but common sample IDs

Chapter 3

Literature Review

Mobility data generate valuable knowledge about human behaviour. They are the cornerstone of many application domains due to the abundance of data emerging from smart devices and sensors. Such domains include traffic estimation (Ny et al. (2014)), crowdsensing (L. Wang, Zhang, Yang, Lim, and Ma (2016)), and other IoT applications (Yin, Xi, Sun, and Wang (2017)).

Accurate mobility data sharing can result in privacy breaches (Zang and Bolot (2011), F. Xu et al. (2017)). For example, an adversary may exploit the uniqueness and regularity of human mobility to recover individual trajectories from location statistics without prior knowledge, as demonstrated in the privacy attack by F. Xu et al. (2017). Moreover, an attacker who knows an individual's home and work locations might infer other location traces, as illustrated in the privacy attack by Zang and Bolot (2011), even when data is anonymized (the person's identity is not released).

Extracting knowledge from mobility data makes the participants vulnerable to membership inference attacks (Zhang et al. (2020)). For instance, an adversary might infer the presence of a victim's data without relying on any prior knowledge of the victim's location (Zhang et al. (2020)). Additionally, the attack presented by L. Zhu, Liu, and Han (2019) shows that it is possible to reconstruct the private training data from the publicly shared intermediate learning parameters (gradients).

A significant amount of work has been done to explore mobility data while preserving the privacy of individuals with differential privacy. We thoroughly reviewed in Errounda and Liu (2019) the privacy applied during mobility data collection and aggregation. This chapter briefly presents differentially private methods to protect raw mobility data. Then, it focuses on methods that enable

differential privacy for mobility data aggregation. Next, we review mobility forecasting methods and differentially private deep learning.

3.1 Differential Privacy for Mobility Data Collection

The privacy mechanism for mobility data collection should prevent an adversary from accurately guessing a user's location or trajectory based on the observed mobility data. This privacy guarantee may be achieved with distance-based, obfuscation-based, and anonymity-based methods.

The idea behind the distance-based method (such as Geo-Indistinguishability (Chatzikokolakis, Palamidessi, and Stronati (2015))) is to restrict the indistinguishability between two locations based on their distance. Therefore, locations within a small distance produce observations with similar distributions, limiting the adversary's knowledge. On the other hand, the user's locations become more distinguishable when at a more significant distance. Consequently, the indistinguishability guarantee (Definition 2) is proportional to the distance between two locations.

Due to its intuitive incorporation of mobility characteristics into differential privacy based on proximity, this approach attracts the interest of researchers generating a rich body of literature. Several works (Ma, Ma, Li, Jiang, and Gao (2017), Chatzikokolakis, Palamidessi, and Stronati (2014), Chatzikokolakis et al. (2015), and Hua, Tong, Xu, and Zhong (2018)) use movement predictability and regularity to protect users' raw mobility data under repeated sharing due to the resulting privacy loss associated with the sequential composition (Theorem 1). For example, the privacy mechanism presented by Ma et al. (2017) uses R-trees to capture the movement regularity; R-tree is a balanced spatial tree index that allows high search performance (Guttman (1984)). The mechanism caches previously released private locations. When the user requires location sharing, the privacy mechanism traverses the R-tree to find a previously released location within the indistinguishability radius. If such a location exists, it is released instead of calculating a new private location. Another privacy mechanism introduced by Hua et al. (2018) divides the location space using a grid and only uses a privacy budget portion to calculate a private location when the user's original location is too close to the grid's cell border.

Finally, the privacy mechanism presented by Chatzikokolakis et al. (2014) and Chatzikokolakis

et al. (2015) predicts the user's location based on historical data to release it in case of a sharing request if it is within the indistinguishability radius. Therefore, these methods optimize the privacy budget usage over continuous location sharing to provide the same level of privacy for more user locations.

Unlike the distance-based method, a distance correlation between the accurate and the perturbed locations is not required in masking-based approaches. A user's location may be represented with a binary vector or a spatial region. For example, each location may correspond to an installed device in an indoor positioning system. Masking-based methods protect the location by applying the randomized response mechanism (as described in Definition 6) on every binary value in the vector (Kim, Kim, and Jang (2018)). The resulting location vector is shared with a data curator.

If the privacy mechanism outputs a region instead of a location, the masking-based method first divides the location space into fine-grained regions (L. Wang et al. (2020), L. Wang et al. (2016), and Q. Yang, Chen, Guizani, and Lee (2021)). Then, it builds a masking matrix that minimizes the expected distance between the user's location region and the masked region. The matrix entry $[i, j]$ corresponds to the probability of masking region r_i with region r_j. Differential privacy binds the maximum probability ratio between any two regions that an adversary observes.

Finally, anonymity-based methods leverage the K-anonymity technique (Gruteser and Grunwald (2003)) to hide the user within a cloaking region of $k-1$ users to prevent a potential adversary from distinguishing between the k users' locations. Anonymity-based methods (X. Yang, Gao, Zheng, and Wei (2020)) use differential privacy to make the cloaking regions indistinguishable. Several techniques are proposed to build the cloaking region. Guo et al. (P. Guo et al. (2021)) propose a reputation system to assess users' honesty based on their previous participation in cloaking regions. A minimum radius determines the cloaking region's shape. When not enough users have a required reputation within the cloaking region, an additional location is calculated by perturbing the region's centroid. Hilbert curve (Hilbert (1935)) is another technique used to build the cloaking region. Used by Ngo and Kim (2015), it preserves the users' proximity when generating the cloaking regions candidates.

3.2 Differentially Private Mobility Data Aggregation

Depending on the application domains, mobility data aggregation may be done once, such as crowdsourcing tasks, or continuously (traffic monitoring).

3.2.1 Singular Mobility Data Aggregate

Several works propose differentially private techniques for publishing mobility aggregates. Most of these techniques target the centralized data setting (N. Li, Yang, and Qardaji (2013), To, Ghinita, and Shahabi (2014), To, Ghinita, Fan, and Shahabi (2017), and Cormode, Procopiuc, Srivastava, Shen, and Yu (2012)) where location data is collected at a single timestamp by a trusted entity, then released with a privacy guarantee. These techniques rely on dividing the location space into smaller areas (cells) and calculating the number of users in each cell. This number helps calculate elaborate aggregates (such as means and medians) or answer requests (such as range queries). The challenge in designing a spatial division method is to optimize the aggregates' utility.

Li et al. (N. Li et al. (2013)) propose a uniform grid partitioning that selects a grid granularity proportional to the total number of locations in the original data, the privacy budget ϵ, and the location distribution uniformity. A challenge arises when answering range queries that cover multiple cells with the uniform grid approach. Avoiding over-partitioning sparse areas or under-partitioning dense areas is crucial since the cumulative noise is proportional to the number of cells involved in the range query. Therefore, the proposed solution includes two granularity levels depending on the users' density.

Another method (Cormode et al. (2012)) uses tree-based partitioning to divide the location space recursively using a tree structure, where each node represents a location region. When answering a range query, the tree is traversed to identify the nodes corresponding to the query's regions, and the number of users is summed, including the perturbation noise. Similar to the grid partitioning method, the global sensitivity used to calibrate the perturbation noise is one.

There are two main types of tree-based partitioning: data-independent and data-dependent. Data-independent decomposition does not consider the distribution of the locations in space. For example, quad-trees recursively split the location space into four equal regions regardless of the

locations in each region. Kd-tree is an example of data-dependent decomposition that recursively splits the location space via lines passing through a median calculated based on the location distribution. However, in the case of data-dependent trees, an adversary can use the median information to gain more knowledge about users' locations. Therefore, noise is also added to the means to make the tree structure differentially private too.

3.2.2 Sequential Mobility Data Aggregate

The user's privacy is compromised when continuously publishing mobility aggregates for monitoring purposes such as traffic analysis and social trend observation. Dwork, Naor, et al. (2010) categorize continuous privacy guarantee approaches into user-level and event-level. Event-level privacy protects several events in the data sequence. Conversely, user-level privacy protects the user in the entire sequence. For example, in privacy solutions for trajectory aggregates, event-level privacy hides the user's participation to a single timestamp in the trajectory's aggregate, the timestamp where the monitored event occurs. On the other hand, user-level privacy hides the user's participation in the aggregates of each timestamp.

The challenge in achieving user-level privacy is the degradation of the privacy guarantee due to the accumulation of privacy budget ϵ at each timestamp. The presented solutions balance user- and event-level privacy by managing the privacy budget distribution. We categorize the solutions into interval-based and sliding window methods. Interval-based methods achieve user-level privacy in the entire mobility stream by selecting timestamps to add noise and approximating aggregates for the in-between timestamps. Sliding window methods guarantee user-level privacy within a window of timestamps that slides over the whole mobility aggregates.

Interval-based Methods

The privacy guarantee quickly degrades when differential privacy is applied continuously. The privacy guarantee decreases by a factor of up to $e^{T\epsilon}$ for T consecutive location collection (see the sequential composition property of differential privacy (Theorem 1)). For example, if we want to protect the users' privacy with an indistinguishability level of $\epsilon = 1$ in a trajectory aggregate composed of 1000 timestamps, we would have to apply differential privacy at each timestamp with

the privacy budget $\frac{\epsilon}{1000} = 0,001$. The problem is that the smaller the privacy budget, the higher the added noise. Therefore, this approach does not provide good utility.

The sample and interpolate solutions solve this problem by choosing timestamps where a portion of the privacy budget is used to calculate a private aggregate. The private aggregates for the in-between timestamps are interpolated. This approach requires answering the following questions:

- What strategy to use to sample the timestamps where private aggregates are calculated?
- How to allocate the privacy budget over the sampled timestamps?
- How to interpolate the in-between aggregates?

The first question can be answered using an interval to sample timestamps to calculate the private aggregate, as adopted by Fan, Xiong, and Sunderam (2013), Fan and Xiong (2014), and Yan et al. (2019). The challenge is in determining the optimal interval size. The cumulative perturbation error introduced at each timestamp increases when the interval's size is small. On the other hand, when the interval size is large, the released aggregates do not reflect up-to-date data values. Fan and Xiong (2014) and Yan et al. (2019) present adaptive interval sampling to solve this problem, where the interval's size changes following the variation in the aggregates. The privacy budget is allocated evenly between all sampling timestamps to answer the second question.

Prediction can be used to answer the third question. The framework introduced by Fan et al. (2013), Fan and Xiong (2014), and Yan et al. (2019) predicts the next aggregate value based on the previously released aggregates at each timestamp. For the in-between timestamps, the predicted aggregates are released. If it is a sampled timestamp, the framework calculates a private aggregate using a portion of the privacy budget. It uses a corrector to reduce the error between the predicted and the private aggregate; the resulting value is then released.

Sliding Window Methods

Interval-based methods require knowing the trajectory's overall length for privacy budget management, which is not always possible. The sliding window methods balance user- and event-level privacy by guaranteeing indistinguishability within a sliding window of w timestamps. This approach requires answering the following questions:

- How to allocate the privacy budget over the timestamps within the sliding window?
- What is the optimal window size?

$w - event$ privacy (Kellaris, Papadopoulos, Xiao, and Papadias (2014), Nie et al. (2016), Liang, Chen, Wu, and Li (2019), and Tudor, Gulisano, Almgren, and Papatriantafilou (2020)) is a sliding window method with w timestamps in each window. The privacy mechanism at each timestamp uses a portion of the privacy budget to calculate a private aggregate. Moreover, the privacy mechanism guarantees that the overall privacy budget in the sliding window is below ϵ.

A fixed window size is not suitable for trajectories that are sparse in time. For example, the locations that need to be privately protected may stretch on a window time larger than w. Therefore, $\ell - trajectory$ (Cao and Yoshikawa (2015)) defines a window size containing ℓ different locations the user visited instead of timestamps.

To adapt the window size, a variable size window is also used by Jo, Jung, and Park (2018) by drawing on traffic data properties, such as road structure and time-based traffic variation. A transition matrix between road sections and intersections models road connectivity. If a road has three consecutive segments, predicting a user's trajectory at the corresponding three sequential timestamps is easy. For example, this property is most relevant for vehicle trajectories. Therefore, the sliding window size is calculated based on the entropy value extracted from the transition matrix. The privacy budget exponentially decreases with each timestamp with a decaying factor determined by the maximum window size possible.

3.3 Mobility Forecasting

Mobility prediction facilitates many real applications, including transportation management and crowd management. For example, it might prevent a potential risk to public safety, such as stampede, during large-scale public events. Deep learning is a pivotal technology in learning the complex spatio-temporal characteristics of mobility data. However, combining neural networks to obtain a precise prediction is challenging. The reviewed work below aims to answer the following questions:

- How to achieve precise mobility forecasting using state-of-the-art spatio-temporal neural networks?
- How to enable the same precision for distributed mobility data through federated learning?
- How to achieve learning when features are distributed vertically over the participating organizations?

This section reviews centralized mobility forecasting to answer the first question and then federated mobility forecasting methods to answer the second. Finally, given our distributed setup (Section 1.1), we review some vertical federated learning methods to answer the third question.

3.3.1 Centralized Mobility Forecasting Learning Models

Mobility data involves a complex correlation between space and location with high regularity and predictability. Therefore, recurrent neural networks (LSTM, GRU) and graph neural networks (CNN and GNN) are prevalent in capturing these aspects for accurate mobility forecasting. The challenge is in finding a method to combine these neural networks.

Several mobility prediction systems seek to anticipate human movement in scenes where they adapt their motion based on the behaviour of other people in the vicinity. For instance, a pedestrian could alter his/her path or stop momentarily to accommodate other people moving toward him/her. It is hard to predict such deviation in trajectory by observing the person in isolation. Therefore, it is difficult to identify trends and seasonality in this data type. The capability of LSTMs to correlate the previous and current information makes them a suitable candidate for mobility prediction.

Several mobility prediction systems blend scene information and human movement trajectories in the prediction process within static crowded scenes (Alahi et al. (2016), Manh and Alaghband (2018), and Xue, Huynh, and Reynolds (2018)). Each pedestrian's mobility is usually considered as a single time sequence data fed to a dedicated LSTM. Social-LSTM (Alahi et al. (2016)) stacks a pooling layer on top of all the LSTMs to correlate movement between pedestrians by sharing the LSTMs' hidden states. Scene-LSTM (Manh and Alaghband (2018)) adopts the same approach and assigns an additional LSTM to each user to incorporate the impact of the scene structures (such as

buildings and static obstacles) on human trajectory prediction. Finally, SS-LSTM (Xue et al. (2018)) assigns one more LSTM to each user to tackle trajectories affected by neighbouring pedestrians.

Prediction mobility is not limited to anticipating a pedestrian's movement; it also aims at acquiring an accurate prediction of crowd flows based on historical data. For instance, crowd monitoring can help event organizers to effectively plan venue layouts, such as food points, shopping points, entries, and exits, and prepare a crowd management strategy, including restricting access, evacuation, and additional installations. Mobility forecasting is also crucial for traffic management. For example, a traffic management department can allocate traffic resources more appropriately depending on the crowd flow or issue appropriate traffic warnings.

CNNs and LSTMs have succeeded in the image processing and sequence prediction areas, respectively. Consequently, more solutions use this combination in spatiotemporal prediction methods. For example, ST-DCCNAL (W. Li et al. (2019)) is a traffic flows forecasting model that captures the spatial correlation by connecting each CNN layer to every other layer in a feed-forward fashion. The features are then sent into an LSTM to discover the underlying temporal dependencies in the data. Similarly, STDN (H. Yao, Tang, Wei, Zheng, and Li (2019)) handles spatial and temporal information via CNN and LSTM, respectively. They use CNN to model the spatial dependency among locations for traffic flow information. Then, an attention mechanism learns the long-term periodic dependency.

Finally, Singh et al. (2020) present a method that neither uses nor identifies trend and seasonality in the crowd counts. Instead, a stack of LSTM networks is used for crowd forecasting to cater to the short-term fluctuations and unpredictable behaviour of large-scale public events with a granularity of forecast horizon taken in minutes. Hence, the learning model obtains future crowd counts based on historical data. Furthermore, CNNs are combined with LSTM to extract the crowd features from the input sequence.

3.3.2 Federated Mobility Forecasting Learning Models

Centralized machine learning methods involve transferring massive amounts of mobility data to a data center to learn a generalized prediction model. However, we have witnessed partnerships

among public agencies and mobile service providers such as Uber and Hellobike. These partnerships require a federated approach to extend the capability and services of companies that provide real-time traffic flow forecasting, traffic management, car sharing, and personal travel applications. For example, frequently transmitting training data and signalling overhead could quickly exhaust the network capacity and negatively impact payload transmissions. With federated learning, these companies could cooperatively train a globally shared model through their local data without exchanging the raw data.

Y. Liu, Zhang, Zhang, and James (2020) propose a GRU-based learning framework (FedGRU) for traffic flow prediction. First, the cloud server distributes a global model copy to all organizations. Each organization trains its copy on local data and shares the encrypted parameters with the cloud server. Once the parameters are aggregated to build a new global model, the cloud server distributes the new global model to each organization. This process repeats until the learning goal is achieved.

Another work (Zhang, Dang, Shihada, and Alouini (2021)) proposes a wireless traffic prediction framework with a model trained collaboratively by multiple organizations. The framework groups the numerous organizations into clusters based on geo-location information and traffic patterns to tackle the data heterogeneity. Then, a global model is trained and shared among the organizations' clusters. Similarly, Lonare and Bhramaramba (2021) use a combination of LSTM and clustering to predict the number of vehicles moving in a direction at a given time on a particular road. The participants train an LSTM that is then shared with the cloud server, which averages all received models to build a global one. Additionally, the server uses a decentralized K-means clustering algorithm to select participants for local training to reduce communication costs by restricting the number of participants. Unfortunately, the spatial correlation between locations belonging to different organizations is not addressed.

3.3.3 Vertical Federated Learning

Vertical federated learning describes the case where data are partitioned vertically by features. Consequently, each learning participant may hold a neural network with a distinctive architecture to train using local data. Moreover, averaging the parameters of each participant to build a global model does not fit this setting.

One approach to train the global is entity resolution. It consists of finding the correspondence between the rows of the training datasets using the sample identifier (Nock et al. (2018)). However, entity resolution techniques are error-prone. Therefore, Nock et al. (2018) study the effect of entity resolution on learning performance considering two organizations that aim to perform logistic regression in the cross-feature space. Hardy et al. (2017) follow the same entity resolution approach for privacy-preserving two-party logistic regression for vertically partitioned data. Secure gradient descent is used to train the logistic regression model, and the intermediate calculated parameters are exchanged between the two participating organizations. Additionally, Romanini et al. (2021) combine Split Neural Networks and Private Set Intersection to train a vertically federated learning algorithm. Split learning divides the model into segments held by different parties or on different devices (Gupta and Raskar (2018)). Each model segment transforms its input data into an intermediate data representation, then transmits it to the next segment until the training process is completed. During backpropagation, the gradient is also propagated across different segments.

Another approach is using ensemble learning to achieve better predictive performance by combining the predictions from multiple models (Ganaie, Hu, Malik, Tanveer, and Suganthan (2022)). Ensemble learning involves multiple models combined in some fashion, such as bagging, stacking, and boosting. Bagging uses decision trees on different samples; stacking uses a meta-model to learn how to best combine the predictions, and boosting adds ensemble networks sequentially that correct the predictions made by prior models through a weighted average.

F. Fu et al. (2021), Le et al. (2021), and Cheng et al. (2021) use boosted decision trees, an ensemble algorithm that trains a series of decision trees as weak learners and enhances the model ability with a boosting strategy. This learning usually involves a scheduler to manage the training routine, dispatch operations to participants, and collect participants' computation results. The scheduler also maintains the decision tree model and the synchronization with the participants. The participants run the operations dispatched from the scheduler on their training data shard and exchange necessary messages with the opposite participants.

3.4 Differentially Private Deep Learning

Deep learning with differential privacy typically involves adding noise to the model's gradients during training to protect each individual's privacy in the training dataset. When all parameters are treated the same regarding the amount of noise injected, it disregards the different impacts these parameters have on the model output. Moreover, a uniform privacy budget may compromise learning, especially when the gradients may be small in proportion to the added noise.

This section reviews two research directions to tackle these challenges. The first trend customizes the noise injected into the deep learning model parameters based on the features' relevance (N. Phan, Wu, Hu, and Dou (2017) and N. Phan et al. (2019)). The second direction explores dynamic privacy budget allocation during the model's training (Gong, Feng, and Xie (2020), Yu, Liu, Pu, Gursoy, and Truex (2019), and Lee and Kifer (2018)).

3.4.1 Feature-Adaptive Learning Algorithms

These solutions inject noise into the learning parameters by assessing the feature's contribution through Layer wise Relevance Propagation (LPR) (Montavon, Binder, Lapuschkin, Samek, and Müller (2019)). LPR operates by propagating the prediction backward in the neural network and computing the relevance of each neuron at each layer based on the messages sent from the previous layer.

The proposed adaptive mechanism (N. Phan et al. (2017)) injects Laplace noise into the computation of LPR to estimate a differentially private relevance of each input feature to the model's output during the preprocessing step using a pre-trained model. Based on the computed feature relevance, the mechanism distributes adaptive noise into the first hidden layer of the model during the stochastic gradient descent using the Laplace mechanism.

Unlike N. Phan et al. (2017), the mechanism presented by Gong, Pan, Xie, Qin, and Tang (2020) adds noise proportional to the feature's relevance to gradients instead of the hidden layer during the backward propagation process. Consequently, more noise is added to gradients whose neurons have less relevance to the model output, while less noise is injected into gradients of neurons that have more relevance to the model output.

The heterogeneous Gaussian mechanism introduced by N. Phan et al. (2019) correlates the feature's relevance to the loss function fluctuation. Consequently, more noise will be injected into less sensitive components, or vice-versa, or even randomly redistributed. Features with higher derivative values in the first hidden layer result in higher loss function calculation. Therefore, they are assigned a smaller privacy ratio to increase their privacy protection. The computation of the feature relevance is achieved during the prepossessing step using a pre-trained model.

Although the proposed algorithms adapt the privacy protection to the feature's relevance, they are designed for a centralized learning setting. Therefore, they assume a single learning model, which is not always possible in the vertical federated learning paradigm. Thus, we cannot adopt them to solve our research problem.

3.4.2 Budget Allocation

Multiple strategies to dynamically allocate the privacy budget are explored by Gong, Feng, and Xie (2020). The interval increase strategy increases the budget to the maximal value with uniform steps. The second strategy exponentially raises the privacy budget, implying that the added noise is reduced slowly in the early epochs and reduced rapidly. A third strategy rapidly decreases the amount of noise in the early epochs and then gradually in the later stages by logarithmically raising the privacy budget. These dynamic allocations are designed for multiple participants in the learning setting; however, the assumption that gradients continuously decrease with each training iteration is not always correct.

Yu et al. (Yu et al. (2019)) propose an algorithm that assigns a privacy budget proportionally to the model's accuracy during the training. The accuracy is assessed periodically using a public validation dataset. Then, the algorithm compares the accuracy difference to a predefined threshold. When the accuracy improves, the privacy budget is increased and remains the same otherwise.

Lee and Kifer (Lee and Kifer (2018)) split the privacy budget at each iteration into two portions. The first portion calculates the optimal learning step to help the gradient descent converge faster. The remaining privacy budget is used to add noise to the gradients.

Although these dynamic allocation methods (Yu et al. (2019) and Lee and Kifer (2018)) do not assume that the gradients consistently decrease during the training, they assume a centralized

learning setting. Therefore, the assumption of a single learning model does not hold in vertical federated learning, where multiple organizations may have different learning model architectures.

Chapter 4

Distributed Release of Mobility Aggregate with Differential Privacy

4.1 Introduction

Mobility data is the cornerstone of many application domains; it is critical to generate valuable knowledge about users' behaviour (Ny et al. (2014), L. Wang et al. (2016), and Yin et al. (2017)). Traffic estimation (Ny et al. (2014)), crowdsensing (L. Wang et al. (2016)), and other IoT applications (Yin et al. (2017)) are examples of these domains. Some applications require a single data collection and offline analysis, such as urban planning. While others, such as intelligent transportation systems, continuously collect and analyze mobility data in real time. However, the collective release of mobility aggregate may result in privacy breaches (X. Yang, Ren, Lin, and Yu (2016)). For instance, an attacker may recover a single user's trajectory solely based on the number of users present at several locations at each time slot, as demonstrated in the privacy attack by F. Xu et al. (2017). Moreover, an attacker who knows a person's work and home locations might infer other location traces, even when data is anonymized (the person's identity is not released) (Zang and Bolot (2011)).

Existing privacy techniques include k-anonymity (Shou, Shang, Chen, Chen, and Zhang (2013)) and encryption (Q. Li, Cao, and La Porta (2014) and Lu, Liang, Li, Lin, and Shen (2012)). K-anonymity (Gruteser and Grunwald (2003)) defines an area that includes k users' locations as a

cloaking region to prevent a potential adversary from distinguishing between the k users' locations. However, this approach is vulnerable to linkage attacks (Krumm (2007)). For example, precomputed cloaked regions may not satisfy the privacy requirement of users due to changes in popularity, resulting in a temporal linkage attack. Furthermore, encryption techniques such as private information retrieval may incur significant communication and computation overheads (X. Yang, Wang, Ren, and Yu (2017)). Moreover, encryption techniques are vulnerable to access pattern attacks. In this case, an attacker may track the user's question and response pairs, then connect them with contextual information to deduce the user's identity (Islam, Kuzu, and Kantarcioglu (2012)).

Differential privacy (Dwork (2008)) is a technique that quantifies the degree to which an individual's privacy is protected when releasing statistics about a dataset. A trusted data curator collects data from multiple individuals in a centralized model, adds carefully selected noise, then shares the final result for analysis. The problem with this model is its vulnerability to privacy attacks, in which an attacker accesses the data gathered by the data curator before data perturbation takes place (Murakami and Kawamoto (2018)).

A distributed derivation of differential privacy is local differential privacy (Duchi et al. (2013)). It enables users to perturb their data before sending it to the untrusted data curator. Local differential privacy has similar guarantees to traditional differential privacy regarding adversaries' side information and composability. Moreover, local differential privacy eliminates the need for a trusted data curator and allows users to customize their privacy level. However, when local differential privacy is used in the continuous setting, the released aggregates' privacy guarantee deteriorates over time (Ding et al. (2017)).

Another challenge in applying differential privacy to distributed released of mobility aggregate is the accumulation of privacy loss with continuous application, which may lead to a poor privacy guarantee. Several solutions exist to address this problem revolving around managing the privacy budget to minimize privacy loss while maintaining the mobility aggregate's utility (Fan et al. (2013), Fan and Xiong (2014), Yan et al. (2019), Kellaris et al. (2014), Nie et al. (2016), Liang et al. (2019), and Tudor et al. (2020)). However, these mechanisms are limited because they are not adapted to local differential privacy.

This chapter investigates the continuous release of mobility aggregates that guarantees local differential privacy. We first define the problem in detail, then lay out the components of our solution.

Following the scenario introduced in Chapter 1, M organizations continuously collect the locations of a group of moving users over a set of N locations. We assume that the location set, N_k, covered by each organization O_k are non-overlapping. Each organization, O_k, builds a real-time mobility data stream, S_t, containing the number of users present at each location. We define a mobility data stream as follows:

Definition 8 (Mobility data stream)**.** *The stream $S_t = \{x_1, x_2, ..., x_t\}$ is a real-time sequence of mobility aggregate where $x_i = \{x_{i,L_1}, ..., x_{i,L_{N_k}}\}$ and x_{i,L_j} corresponds to the number of users present at a location L_j at timestamp i.*

A central server, $\mathcal{C}$, is responsible for gathering every mobility data stream for real-time monitoring or building a historical dataset of the users' movement. The trust boundary of each organization excludes $\mathcal{C}$ and the other organizations. Moreover, we assume that the communication between the organizations and $\mathcal{C}$ is secure. This setting results in heterogeneous privacy needs since each organization separately aims to protect its mobility data stream from privacy leaks. Hence, we introduce the first research question:

> RQ1: How can each organization enable differential privacy locally to protect its mobility data stream?

The application of differential privacy to release the mobility data stream can be seen as querying a static dataset of mobility counts the following question: "What is the number of users at each location?" at each timestamp i. Consequently, random noise calibrated using a privacy budget, ϵ_i, needs to be added at each timestamp, i, in the mobility data stream to protect the privacy of the moving individuals. Based on the composition Theorem 1, the overall privacy loss of the mobility data stream is the sum of the privacy budgets at each timestamp. However, in a real-time monitoring setting, assessing the length of the mobility data stream to derive the privacy budget portion applicable at each timestamp is impossible. Consequently, we must answer the following.

RQ2: How to dynamically allocate the privacy budget to prevent mobility data stream's privacy leaks?

Another consequence of the repeated application of differential privacy is error accumulation stemming from the random noise addition, which may lead to poor utility. Naturally, the following question arises:

RQ3: How to minimize the difference between the accurate mobility aggregate and the private release while guaranteeing differential privacy?

Finally, differential privacy introduces enough random noise into the original mobility aggregate to guarantee the moving individual's privacy. However, we also need a method to formally assess the effect of the budget allocation on the aggregate's accuracy. Therefore, we answer the following.

RQ4: What is the theoretical upper bound between the accurate mobility aggregate and the private release?

This thesis answers the question presented above as follows:

- We adopt the randomized response algorithm and a similarity threshold between mobility aggregates to enable differential privacy locally to each organization (RQ1). We present the detailed algorithm in Section 4.2.1.
- We follow a sliding window approach to allocate the privacy budget dynamically (RQ2). We determine a budget at each timestamp by balancing the budgets assigned in the previous window and the remaining timestamps in the current window (Section 4.2.2).
- We minimize the difference between the accurate mobility aggregate and the private release by reusing the previously released aggregates (RQ3). We adopt the exponential mechanism to guarantee that this strategy provides a differential privacy guarantee (Section 4.2.3).
- We leverage the widely used α, β-usefulness metric (Section 4.3) to determine the theoretical upper bound between the accurate mobility aggregate and the private release (RQ4).

We use two real-world datasets to evaluate our solution (a real-time proximity detection data stream deployed by a private telecommunication company in Montreal and a stream of bike numbers statistics of bike stations *Capital Bike Share* (2016)). We compare our proposed solution with state-of-the-art algorithms (Kellaris et al. (2014)), and the results show the mean absolute error decrease by up to 19% (Section 4.4).

4.2 Distributed Release of Mobility Aggregate with Differential Privacy

We propose an aggregate sharing method using a sliding window. Following the intuition presented by Kellaris et al. (2014), we define a window that includes w timestamped aggregate releases with a total privacy budget of ϵ. However, we improve the utility of the shared aggregate through a combination of dynamic budget allocation and approximations. Therefore, a sliding window of size w start at timestamp t and ends at timestamp $t + w - 1$.

We determine the private statistics based on the accurate mobility aggregate through three steps: decision, perturbation, and approximation. As illustrated in Figure 4.1, at every timestamp t, each organization calculates the mobility aggregate (by counting the users present in each location). Then, it compares the accurate aggregate locally with the last private release. The similarity test aims to compare the more efficient approach: approximate the current aggregate with the previous release, or publish them with the necessary noise. The approximation strategy returns the closest private release if the aggregates are similar. Otherwise, a privacy budget is assigned to perturb the accurate aggregate before release.

Algorithm 1 presents the details of the three steps for the private statistic release. We summarize the used notations in Table 4.1. Algorithm 1 takes as input: (1) the total privacy budget ϵ, (2) the accurate location statistic x_t, (3) the last private release $x_l{}'$, and (4) the similarity threshold T. Since we access the private aggregate at each step, we must ensure that each access's outcome remains differentially private for the algorithm to satisfy differential privacy. Moreover, the three steps are not guaranteed to be executed within the same memory block. For example, an adversary might capture the outcome decision step result and infer the accurate aggregate. Therefore, the privacy

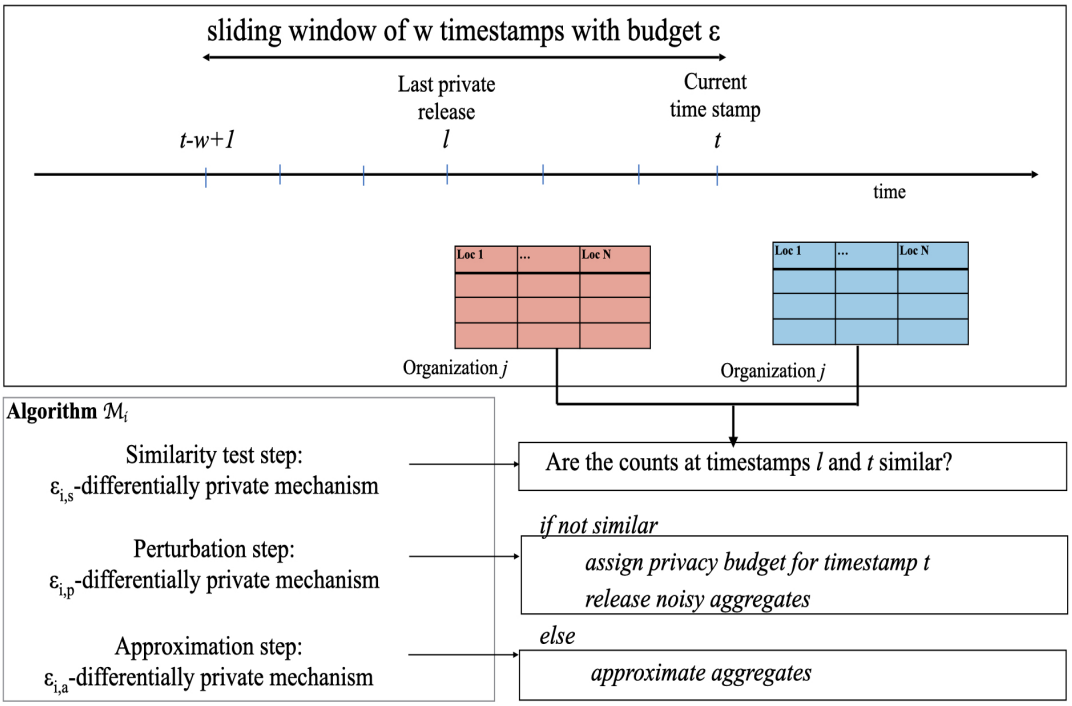


Figure 4.1: The three steps of the collective local differential privacy algorithm: the similarity step compares the previous and current mobility aggregate, the approximation strategy returns the closest private release, and the perturbation step computes a privacy budget to calibrate the added noise to the accurate aggregate.

budget allocated to the current timestamp t is ϵ_t. It is divided between the three steps as follows $\epsilon_t = {\epsilon_t}^d + {\epsilon_t}^p + {\epsilon_t}^a$. Finally, the overall privacy budget used for the sliding window is ϵ.

Notation	Description
x_t	Accurate location statistic at current timestamp t
${x_t}'$	Private location statistic at timestamp t
l	Last private release timestamp l
w	Sliding window size
ϵ	Overall privacy budget
ϵ_t	Privacy budget allocated at the current timestamp
${\epsilon_t}^d$	The decision step privacy budget
${\epsilon_t}^p$	The perturbation step privacy budget
${\epsilon_t}^a$	The approximation step privacy budget
${\epsilon_t}^f$	Estimated forward budget at timestamp i
${\epsilon_t}^b$	Estimated backward budget at timestamp i

Table 4.1: Used notations for distributed release of mobility aggregate with differential privacy

We divide the overall budget of the sliding window ϵ into three parts corresponding to the three

steps. Since the higher the budget, the smaller the distance between the true location statistic and the perturbed one, we allocate the bulk of the privacy budget to the perturbation step: $\epsilon/2$. The budget assigned to the decision and the approximation step is used to either: (1) randomly answer the similarity test or (2) select the closest private release with differential privacy guarantee. These steps do not add no perturbation, and thus they require less privacy budget. We assign the budget $\epsilon/4$ equally to the decision and the approximation step.

Algorithm 1 The local differential privacy algorithm $\mathcal{M}_t$ at the current timestamp t

Require: Current mobility aggregate x_t, last mobility aggregate release $x_l{}'$, similarity threshold T, sliding window size w, total privacy budget ϵ

Ensure: Private statistic $x_t{}'$

1: ${\epsilon_t}^d \leftarrow \epsilon/4 * w$

*//** The decision step **//*

2: Calculate the similarity between x_t and $x_l{}'$ given the threshold T using ${\epsilon_t}^a$ privacy budget

3: **if** x_t and $x_l{}'$ are similar **then**

*//** The approximation step **//*

4: ${\epsilon_t}^a \leftarrow \epsilon/4 * w$

5: $x_t{}' \rightarrow$ Approximation strategy $\mathcal{M}_1$ with privacy budget ${\epsilon_t}^a$

6: **else**

*//** The perturbation step **//*

7: ${\epsilon_t}^p \leftarrow$ Privacy budget allocation method

8: $x_t{}' = x_t + Noise({\epsilon_t}^p)$

9: **end if**

4.2.1 Decision Step

A dataset D_t is formed at each timestamp t, with d attributes. Each row in D_t corresponds to an individual user. The decision step examines how similar the current and the last privately released aggregates are. A new private aggregate computation is needed if the aggregates are not similar. MAE values determine the statistical similarity calculated below:

$$MAE = \|x_t - x_l{}'\| \tag{12}$$

where the vector x_t denotes the aggregates computed from the dataset D_t at timestamp t. Timestamp l is where private aggregates $x_l{}'$ are last calculated.

The decision step takes as input: (1) the current mobility aggregate x_t, (2) the last private release $x_l{}'$, and (3) the comparison threshold T. It begins with the comparison of x_t and $x_l{}'$ with regards to the threshold T. The randomized response mechanism is then applied to the similarity result using a portion of the privacy budget ${\epsilon_t}^d$. Recall in Chapter 2 that the randomized response satisfies local differential privacy. This property ensures that the decision step guarantees local differential privacy. Unlike the existing w-event privacy algorithms that require calculating the MAE of the dataset involving all users to perform the similarity test (Equation 12), our solution runs the similarity test locally for every single user. The decision step consumes a budget of $\epsilon/4 * w$ at each timestamp.

The algorithm $\mathcal{M}_t$ at timestamp t is the sequence of the decision algorithm $\mathcal{M}_{1,t}$ and the perturbation algorithm $\mathcal{M}_{2,t}$. The decision algorithm $\mathcal{M}_{1,t}$ is defined in Algorithm 2. It takes the inputs as the current location statistic x_t, the last private release $x_l{}'$, and the comparison threshold T. It starts by comparing x_t and $x_l{}'$ with respect to T. Then the randomized response mechanism is executed on the result of the similarity calculation.

Algorithm 2 The decision algorithm $\mathcal{M}_{1,t}$

Require: Current mobility aggregate x_t, last mobility aggregate release $x_l{}'$, similarity threshold T
Ensure: Similarity result ${v_t}'$

1: Calculate similarity: $T_t = x_t - x_l{}'$
2: $v_t = (T_t > T)$
3: ${v_t}' = \begin{cases} 1, & \text{with probability } p \\ 0, & \text{with probability } p \\ v_i & \text{with probability 1 - 2} * p \end{cases}$

4.2.2 Perturbation Step

If the similarity is beyond the threshold T, the perturbation step uses ${\epsilon_t}^p$, a privacy budget piece, to perturb the accurate mobility aggregate x_t. First, we present the privacy budget allocation strategy used to calculate ${\epsilon_t}^p$. Then, we review the two random noise generation algorithms that can be used for the mobility aggregate perturbation.

Algorithm 3 presents the steps of the privacy budget allocation method. The algorithm takes as input: (1) the previously allocated budget within w window $(\epsilon_1, ..., \epsilon_{t-1})$ and (2) last private release timestamp l. We determine the optimal privacy budget value to allocate at the current timestamp t based on two standpoints: forward and backward sliding window projections. Figure 4.2 illustrates an example of the privacy budget allocation. The forward sliding window starts at the last private release l, while the backward window at timestamp $t + 1 - w$. Both sliding windows contain w timestamps. From the standpoints of the two windows, we compute the privacy budget. Then, to avoid depleting the privacy budget, we choose the smallest of the two values.

The random noise generation mechanism uses the privacy budget ${\epsilon_t}^p$, allocated to timestamp t, to calibrate the noise added to the accurate statistics. Two different probability distributions may be used in the perturbation mechanism; the Laplace distribution and the staircase distribution. The staircase algorithm is an alternative to the Laplace mechanism, where the functional form of the distribution is optimized to minimize the noise variance. The distribution shape is a piecewise discontinuous step function, tapering off geometrically on both sides. We explore in the experiment section the impact of the different random noise generation mechanisms on the collective local differential privacy algorithm's utility.

Algorithm 3 The perturbation step privacy budget allocation $\mathcal{M}_{t,2}$

Require: Previously allocated budget within w window $(\epsilon_1, ..., \epsilon_{t-1})$, last private release timestamp l

Ensure: Allocated budget ${\epsilon_t}^p$

//Forward sliding window projection

1: Compute the timestamps remaining in the forward sliding window projection $r = w - t + l$

2: Compute the privacy budget left for the forward sliding window projection $\epsilon_{remaining} = \epsilon/2 - \epsilon_l$

3: Compute the estimated forward budget ${\epsilon_t}^f = \epsilon_{remaining}/r$

//Backward sliding window projection

4: Compute the privacy budget left for the backward sliding window project $t + 1 - w$: $\epsilon_{remaining} = \epsilon/2 - \sum_{j=t-w+1}^{t-1} \epsilon_j$

5: Compute the estimated backward budget ${\epsilon_t}^b = \epsilon_{remaining}/2$

6: Return the minimum of the two estimated budgets ${\epsilon_t}^p = min({\epsilon_t}^f, {\epsilon_t}^b)$

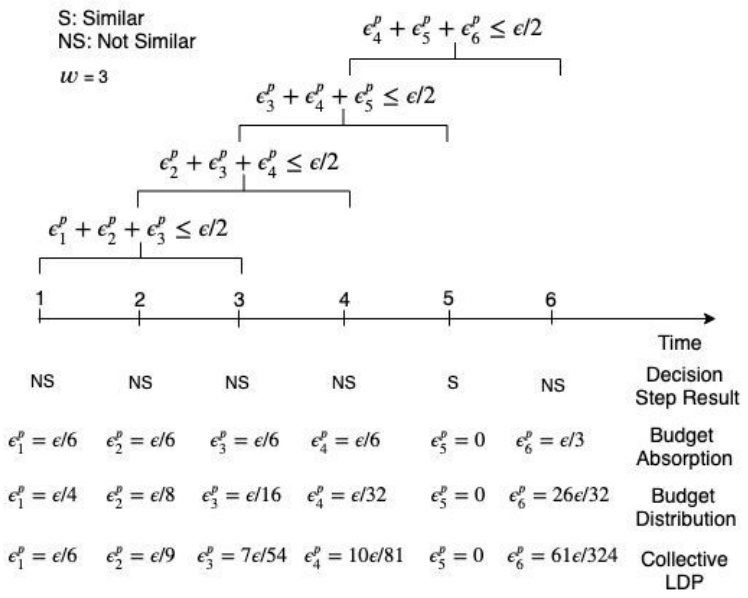


Figure 4.2: We illustrate the privacy budget allocation with an example, where w=3 and 6 timestamps. Assume that the decision step returns a dissimilarity for timestamps 1, 2, 3, 4, and 6. At timestamps 5, the statistics are similar. Therefore, the released statistics is approximated, and no budget is used for the perturbation step. For the first timestamp 1, the last privacy budget allocated is 0, and the forward sliding window projection has w timestamps. Consequently ${\epsilon_1}^f = \epsilon/6$. Also, the backward sliding window did not consume any privacy budget, then ${\epsilon_1}^b = \epsilon/4$. The minimum value of ${\epsilon_1}^b$ and ${\epsilon_1}^f$ is assigned to $\epsilon_i{}^p = \epsilon/6$. The same logic is used for timestamps 2, 3, and 4. We perform the same calculation for the budget distribution and the budget absorption approaches. For all the approaches, the privacy budget within a window of 3 timestamps never exceeds $\epsilon/2$.

4.2.3 Approximation Step

Releasing the last private mobility aggregate $x_l{}'$ has the advantage of not using any privacy budget as it maximizes the privacy budget values for the perturbation step, optimizing the privacy budget consumption. However, this approximation strategy may lead to utility loss.

For example, let us consider consecutive timestamps t and $t+1$ with mobility aggregates similar to the last private release $x_l{}'$, meaning that the original aggregates are within the interval $|x_l{}' - T, x_l{}' + T|$. The released mobility aggregate stays the same ($x_l{}'$), even if the aggregates have changed by up to $|x_t - x_{t+1}| \simeq 2T$. Therefore, this approach s in a loss of utility by hiding a significant change in the accurate aggregates.

To solve this problem, we select the aggregates $x_j{}'$ that is closest to the accurate mobility aggregate x_t. Our solution uses the exponential mechanism (McSherry and Talwar (2007)). The details

of our new exponential approximation approach are described in Algorithm 4. The key idea is to store private releases. Then, we approximate the released private aggregate with the closest private release to the accurate mobility aggregate. This approximation requires retrieving the accurate aggregate to assess the closest private release. Therefore, we must dedicate a privacy budget to the approximation strategy to guarantee differential privacy.

The algorithm takes as input: (1) the accurate mobility aggregate at current timestamp x_t, (2) the previous releases ${x_j}'j \in \{1, l\}$, (3) the similarity threshold T, and (4) the privacy budget ${\epsilon_t}^a$. The notations are defined previously in Table 4.1. We choose the exponential mechanism with a score function u designating the distance between the current mobility aggregate and the previous private releases. Adding or deleting a user modifies the counts by at most 1. Hence, the utility function u has a global sensitivity of one.

Algorithm 4 The exponential approximation strategy $\mathcal{M}_{3,t}$

Require: Current mobility aggregate x_t, historical mobility aggregate releases ${x_j}'j \in \{1, l\}$, similarity threshold T, approximation budget ${\epsilon_t}^a$

Ensure: Approximated release ${x_t}'$

1: **if** $\nexists\, {x_j}', j \in \{1, l\}$ such that ${x_j}' \in [x_t - T, x_t + T]$ **then**
2: return ${x_l}'$
3: **else**
4: **for** ${x_j}', j \in \{1, t\}$ such that ${x_j}' \in [x_t - T, x_t + T]$ **do**
5: Calculate the score function $u \rightarrow |x_t - {x_j}'|$
6: Allocate probability as $exp({\epsilon_t}^a u/2)$
7: **end for**
8: return ${x_t}'$ with probability $\propto exp({\epsilon_t}^a u/2)$
9: **end if**

4.3 Formal Privacy and Utility Analysis

This section presents the formal analysis of Algorithm 1 from the privacy and utility perspectives.

4.3.1 Privacy Analysis

Algorithm 1 is composed of three steps: decision (Algorithm 2), perturbation (Algorithm 3), and approximation (Algorithm 4). To prove that Algorithm 1 guarantees ϵ-differential privacy in any sliding window, we need to prove that each step individually provides the same guarantee.

Let us consider the mobility data stream $S_t = \{x_1, x_2, ..., x_t\}$. Guaranteeing ϵ-differential privacy in any sliding window means that the summation of the privacy budgets for algorithms 2, 3, and 4 on the total timestamps is below ϵ. Below, we prove that Algorithms 2, 3, and 4 satisfy differential privacy on any sliding windows of S_t with the privacy budgets $\epsilon/4$, $\epsilon/2$, and $\epsilon/4$, respectively. Then, we prove that Algorithm 1 guarantees ϵ-differential privacy in any sliding window on S_t.

Theorem 3. *The decision step (algorithm 2) satisfies differential privacy with the budget $\epsilon/4$ on any sliding windows of the mobility data stream S_t.*

Proof. First, we prove that the algorithm is differentially private at each timestamp i. Then, we demonstrate the same guarantee on any sliding window. Let v_i be the result of comparing the aggregate mobility similarity between the current timestamp, x_i, and the last release, $x_l{}'$. When the aggregates are similar ($v_i = 1$), the probability of $\mathcal{M}_{1,i}$ outputting a corresponding similarity result, $v_i' = 1$, is the sum of $1 - 2 * p$ and p. While the probability of outputting a converse similarity result, $v_i' = 1$, is p. Therefore,

$$\frac{Pr[v_i' = 1 \mid v_i = 1]}{Pr[v_i' = 1 \mid v_i = 0]} = \frac{p + 1 - 2 \times p}{p} = \frac{1 - p}{p}$$

Applying similar reasoning to dissimilar aggregates, we obtain the following:

$$\frac{Pr[v_i' = 0 \mid v_i = 0]}{Pr[v_i' = 0 \mid v_i = 1]} = \frac{1 - p}{p}$$

Therefore, $M_{1,i}$ is $ln\langle\frac{1-p}{p}\rangle$-differentially private because the degree of indistinguishability between the two possible inputs "0" and "1" given an observed output v_i' is $ln\langle\frac{1-p}{p}\rangle$. According to the sequential composition theorem (Theorem 1), for any sliding window of size w, and at any timestamp i in the stream S_t, the privacy budget consumed by $\mathcal{M}_1$ within the sliding window is the sum of the privacy budgets of $\mathcal{M}_{1,k}$ such that $i - w + 1 \leq k \leq i$. Therefore the privacy budget of $\mathcal{M}_1$ within

a sliding window w is:

$$\frac{\epsilon}{4} = \sum_{k=i-w+1}^{k=i} ln(\frac{1-p}{p}) = w \times ln(\frac{1-p}{p})$$

where p is $\frac{1}{e^{\frac{\epsilon}{4w}}+1}$ Therefore, $\mathcal{M}_1$ satisfies differential privacy with the privacy budget $\epsilon/4$, which concludes the proof. □

Theorem 4. *The perturbation step (Algorithm 3) satisfies differential privacy with the budget of* $\epsilon/2$ *on any sliding windows of the mobility data stream* S_t.

Proof. The perturbation step (Algorithm 3) is composed of the sequence $\{\mathcal{M}_{2,1},..., \mathcal{M}_{2,t}\}$, such that $\mathcal{M}_{2,i}$ is a mechanism that perturbs x_i at timestamp i. To show that Algorithm 3 provides $\epsilon/2$-local differential privacy on all sliding windows of w timestamps, we first show that each $\mathcal{M}_{2,i}$ is ${\epsilon_i}^p$-differentially private. Then, we demonstrate that the sum of ${\epsilon_i}^p$ within a w sliding window is below $\epsilon/2$.

Since a mobile user can be at only one location at a time, the change in the mobility aggregate, x_i (number of mobile users at timestamp i), is at most 1. Hence, the global sensitivity (Equation 4) is 1, and applying Laplace noise with the scale $1/{\epsilon_i}^p$, as described in Definition 3 makes the algorithm $\mathcal{M}_{2,i}$ ${\epsilon_i}^p$-differentially private.

Now, we show that $\sum_{j=i-w+1}^{j=i} {\epsilon_i}^p \leq \epsilon/2$, for every timestamp i.

$$\begin{aligned}\sum_{j=i-w+1}^{j=i} {\epsilon_j}^p &= \sum_{j=i-w+1}^{j=i} min(\epsilon_j^f, \epsilon_j^b) \\ &\leq min(\underbrace{\sum_{j=i-w+1}^{j=i} \epsilon_j^f}_{\text{L}}, \underbrace{\sum_{j=i-w+1}^{j=i} \epsilon_j^b}_{\text{R}})\end{aligned}$$

Because $i+1-w$ is the last private release, we substitute l with $i+1-w$ and i with $2-w+i$ in

Algorithm 3. We obtain:

$$\sum_{j=i-w+1}^{j=i} \epsilon_j^f = \epsilon_{i-w+1} + \sum_{i-w+2}^{i} (\epsilon/2 - \epsilon_{i-w+1})/(1+w)$$
$$= \epsilon_{i-w+1} + (w-1) \times (\epsilon/2 - \epsilon_{i-w+1})/(1+w)$$
$$\leq \epsilon_{i-w+1} + (\epsilon/2 - \epsilon_{i-w+1}) = \epsilon/2$$

Therefore $L = \sum_{j=i-w+1}^{j=i} \epsilon_j^f \leq \epsilon/2$.

Similarly, we have:

$$\sum_{j=i-w+1}^{j=i} \epsilon_j^b = \epsilon/2 - \sum_{j=i-w+1}^{j=i} \epsilon_{i-w+1}/2^j$$

Suppose that, without loss of generality, the maximum budget value ($\epsilon/2$) is available at timestamp $1-w+i$. Based on line 5 in Algorithm 3, $\epsilon_{i-w+1} = (\epsilon/2) \times (1/2)$. Therefore:

$$\sum_{j=i-w+1}^{j=i} \epsilon_j^b = \sum_{j=1}^{j=w} (\epsilon/2) \times (1/2^j) = \epsilon/2 \times \sum_{j=1}^{j=w} (1/2^j)$$
$$= \epsilon/2 \times (1 - 1/2^w) \leq \epsilon/2$$

Consequently, $R = \sum_{j=i-w+1}^{j=i} \epsilon_j^b \leq \epsilon/2$ and:

$$\sum_{j=i-w+1}^{j=i} {\epsilon_j}^p \leq min(\sum_{j=i-w+1}^{j=i} \epsilon_j^f, \sum_{j=i-w+1}^{j=i} \epsilon_j^b) \leq \epsilon/2$$

□

Theorem 5. *The approximation step (Algorithm 4) satisfies differential privacy with the budget of $\epsilon/4$ on any sliding windows of the mobility data stream S_t.*

Proof. The approximation algorithm (Algorithm 4) on S_t is composed of the sequence $\{\mathcal{M}_{3,1}$,..., $\mathcal{M}_{3,t}\}$ such that $\mathcal{M}_{3,i}$ is a mechanism that perturbs x_i at timestamp i. We apply the exponential algorithm at each $\mathcal{M}_{3,i}$ with the budget ${\epsilon_i}^a = \epsilon/4 * w$. According to Definition 4, every $\mathcal{M}_{3,i}$

satisfies local differential privacy with the budget ${\epsilon_i}^a$.

According to Theorem 1, Algorithm 4 uses the privacy budget $\sum_{i-w+1}^{i} {\epsilon_i}^a$ within a w sliding window. Since we allocate $\epsilon/4 * w$ to the privacy budget for the approximation strategy, then $\sum_{i-w+1}^{i} {\epsilon_i}^a = \epsilon/4$. We conclude that Algorithm 4 satisfies differential privacy with the budget of $\epsilon/4$ on any sliding windows of the mobility data stream S_t. □

Theorem 6. *The collective local differential privacy algorithm (Algorithm 1) guarantees ϵ-local differential privacy on any sliding windows of the mobility data stream S_t.*

Proof. We apply the sequential composition theorem (Theorem 1) to the three steps in the collective local differential privacy algorithm. Therefore, the privacy budget used by Algorithm 1 is the addition of the privacy budgets used by the similarity step (Algorithm 2), the perturbation step (Algorithm 3), and the approximation step (Algorithm 4). Based on theorems 3, 4, and 5, the similarity, perturbation, and approximation algorithms satisfy differential privacy on all sliding windows of w timestamps with the budgets $\epsilon/4$, $\epsilon/2$, and $\epsilon/4$, respectively. Hence, Algorithm 1 provides differential privacy on all sliding windows of w timestamps with a budget of $\epsilon = \epsilon/4 + \epsilon/2 + \epsilon/4$. □

4.3.2 Utility Analysis

To formally determine the theoretical upper bound between the accurate mobility aggregate, x_i, and the private release, ${x_i}'$, we define the following distance error DE:

$$DE = x_i - {x_i}' \tag{13}$$

We leverage the widely used utility definition, provided by Blum, Ligett, and Roth (2013), to show that Algorithm 1 is (α,β)-useful w.r.t. DE for every timestamp in the sliding window. This utility metric defines a mechanism as useful when its output (perturbed aggregate) approximates well the accurate aggregate. Formally:

Definition 9 ((α,β)-usefulness (Blum et al. (2013))). *A privacy mechanism $\mathcal{M}$ is (α,β)-useful w.r.t a query q if $\mathcal{M}$ meets the following for every dataset X and $\hat{X} = \mathcal{M}(X)$, with a probability of at*

least 1-β:

$$q(\hat{X}) - q(X) \leq \alpha \tag{14}$$

Since the private release, $x_i{}'$, is the result of either a perturbation or an approximation, we must show that Algorithm 1 is (α,β)-useful for both cases.

Lemma 1. *Suppose a timestamp i where the mobility aggregate is perturbed. The difference, DE, between the true aggregate, x_i and the private release, $x_i{}'$, is less than α with a probability of at least $1 - \beta$:*

$$Pr[DE \leq \alpha] > 1 - \beta$$

Here, $\alpha \geq -\frac{ln(2*\beta)}{\epsilon_i}$.

Proof. The noise added at each timestamp i is proportional to Lap($\frac{1}{\epsilon_i}$). Therefore:

$$Pr[Lap(\frac{1}{\epsilon_i}) > \alpha] = \int_{\alpha}^{\infty} \frac{\epsilon_i}{2} e^{-x*\epsilon_i} dx$$

Let $\beta = \int_{\alpha}^{\infty} \frac{\epsilon_i}{2} e^{-x*\epsilon_i} dx$, then:

$$\begin{aligned}
\int_{\alpha}^{\infty} e^{-x*\epsilon_i} dx &= \frac{2*\beta}{\epsilon_i} \\
-\frac{1}{\epsilon_i} e^{-x*\epsilon_i} \Big|_{\alpha}^{\infty} &= \frac{2*\beta}{\epsilon_i} \\
-e^{-\alpha*\epsilon_i} &= 2*\beta \\
\alpha &= -\frac{ln(2*\beta)}{\epsilon_i}
\end{aligned}$$

□

Therefore, when $\alpha \geq -\frac{ln(2*\beta)}{\epsilon_i}$, the error DE induced by Algorithm 1, at a timestamp where the location statistic is perturbed is limited within α with high probability (1-β).

Lemma 2. *Suppose a timestamp i where statistics are approximated. The difference, DE, between*

the true aggregate, x_i and the private release, $x_i{}'$, is less than α with a probability of at least $1-\beta$:

$$Pr[DE \leq \alpha] > 1 - \beta$$

Here, $\alpha > \frac{T}{\beta}$.

Proof. DE depends on the aggregate $x_i{}'$ selected by the exponential approximation algorithm (Algorithm 4). Let $\{x_1{}', x_2{}', ..., x_m{}'\}$ be the m previously released private aggregates such that $x_k{}' \in [x_i - T, x_i + T]$ for every $k \in \{1, .., m\}$. Algorithm 4 selects an aggregate, $x_k{}'$, with probability $exp(\epsilon_i{}^a |x_i - x_k{}'|/2\Delta u)$ with $k \in \{1, .., m\}$. Given that the score function sensitivity is one and our privacy budget allocation scheme, $\epsilon_i{}^a$ is $\epsilon/4 * w$. Based on *Markov's inequality*, we have:

$$\frac{\mathbb{E}[DE]}{\alpha} \geq Pr[DE > \alpha]$$
$$\Rightarrow 1 - \frac{\mathbb{E}[DE]}{\alpha} < Pr[DE \leq \alpha]$$

Next, we evaluate $\mathbb{E}[DE]$.

$$\mathbb{E}[DE] = \sum_{k=1}^{k=m} (|x_i - x_k{}'|) * exp(\frac{\epsilon * |x_i - x_k{}'|}{8 * w}/n)$$

n is the probability normalization factor $\sum_{k=1}^{k=m} exp(\frac{\epsilon * |x_i - x_k{}'|}{8 * w})$. For all $k \in \{1, .., m\}$, we have $[x_i - T, x_i + T]$. Therefore, the maximum distance between x_i and $x_k{}'$ with $k \in \{1, .., m\}$ is T. Then:

$$\mathbb{E}[DE] \leq T$$
$$Pr[DE \leq \alpha] > 1 - \frac{T}{\alpha}$$

Let $1 - \frac{T}{\alpha} > 1 - \beta$, then $\alpha > \frac{T}{\beta}$, which concludes the proof. □

Based on lemmas 1 and 2, we can conclude the following theorem:

Theorem 7. *For all sliding windows of size w, for all timestamps i where the statistics are perturbed, we have $Pr[DE \leq \alpha] > 1 - \beta$ for $\alpha \geq -\frac{ln(2*\beta)}{\epsilon_i}$, and for all timestamps i where the*

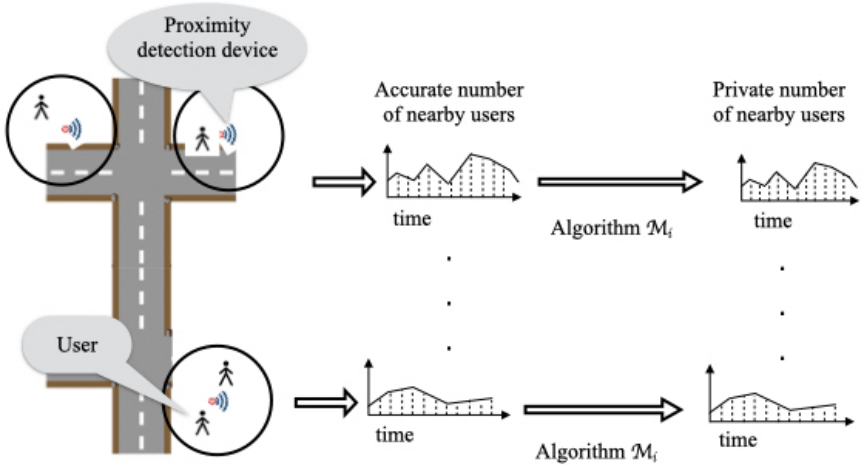


Figure 4.3: Telecommunication dataset: Proximity detection devices capture the number of nearby users at each timestamp. To protect the participants' privacy, the collective differential privacy algorithm (Algorithm 1) is applied at each device.

statistics are approximated, we have $Pr[DE \leq \alpha] > 1 - \beta$ *for* $\alpha > \frac{T}{\beta}$. *Therefore, Algorithm 1 is* (α,β)*-useful.*

4.4 Experimental Evaluation

We evaluate our algorithms' utility on two real-world datasets using multiple metrics. We design several experiments that isolate and compare algorithms 2, 3, and 4 performances to state-of-the-art solutions. The source code used to realize our empirical evaluation is publicly shared [1].

4.4.1 Datasets and Metrics

The first dataset we use is a real-time proximity detection system deployed by a private telecommunication company in Montreal. As illustrated in Figure 4.3, three proximity detection devices are deployed in downtown Montreal. The devices continually gather data about users within their connectivity range through Wi-Fi. Then, each device calculates the total number of nearby users. The overall time for data collection is 1 hour, and it contains 6054 timestamps.

The second dataset is a popular bicycle-sharing system in Washington, D.C. *Capital Bike Share* (2016). As shown in Figure 4.4, the dataset captures trip details containing bike numbers, start

[1] https://github.com/FatimaErrounda/CollectiveLocalDifferentialPrivacy.git

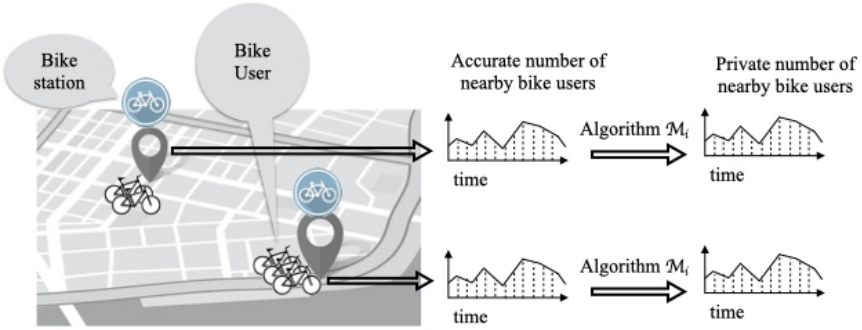


Figure 4.4: Bikeshare dataset: The collective local differential privacy algorithm (Algorithm 1) is applied to the number of bike owners at each bike station. Then, the resulting private number of bike owners is shared.

and end stations, and start and end timestamps. The Bikeshare dataset is published quarterly; we use the first four months of 2016, which contain 3817 bike trajectories. We transform them into a dataset of 368 bike stations with the number of bikes stationed at each timestamp resulting in 52034 timestamps.

We compare the private and the accurate aggregates using the following metrics: Mean Absolute Error (MAE), Relative Mean Error (RME), and Root Mean Square Error (RMSE), defined in Equations (15), (16), and (17). MAE and RMSE measure the degree of variation of a changing quantity. They indicate the difference between the accurate mobility aggregate and the private release without factoring in the accurate aggregate. However, RMSE is more sensitive to higher error values because the error is squared. Therefore, we capture RME, which is more sensitive to the original aggregate value. Indeed, RME multiplies the aggregate with its difference with the private release.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x_i'| \tag{15}$$

$$RME = \frac{1}{n}\sum_{i=1}^{n}|(x_i - x_i') * x_i| \tag{16}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x_i')^2} \tag{17}$$

where x_i is the accurate mobility aggregate, x_i' is the private release, and n is the number of aggregates.

4.4.2 Comparison Frameworks

We compare our algorithms to the state-of-the-art from three perspectives: random noise mechanism, privacy budget allocation, and approximation approach:

- **Random noise mechanism:** Once Algorithm 3 determines the privacy budget, ϵ_i^p, allocated to the timestamp i, it calibrates the added noise using ϵ_i^p. We introduced the Laplace mechanism (Definition 3) as one of the most commonly used mechanisms. The staircase mechanism is another noise generation algorithm presented by Geng and Viswanath (2016) to optimize the Laplace mechanism w.r.t utility by a piecewise representation. It is defined as follows:

  Definition 10 (Staircase Mechanism (Geng and Viswanath (2016))). *The staircase algorithm has the following probability density function:*

$$f_\gamma(x) = \begin{cases} a(\gamma), & x \in [0, \gamma\Delta q) \\ e^{-\epsilon}a(\gamma), & x \in [\gamma\Delta q, \Delta q) \\ e^{-k\epsilon}f_\gamma(i - k\Delta q), & x \in [k\Delta q, (k+1)\Delta q) \textit{ for } k \in \mathbf{N} \\ f_\gamma(-x), & x < 0 \end{cases}$$

  where $a(\gamma) \stackrel{\text{def}}{=} \frac{1-e^{-\epsilon}}{2\Delta q(\gamma + e^{-\epsilon}(1-\gamma))}$, *$\Delta q$ is the global sensitivity.*

  $a(\gamma)$ is set to make sure that integrating $f_\gamma(x)$ over all real numbers results in a value of 1; in other words, the area under the curve must be equal to 1 (Geng and Viswanath (2016)).

- **Privacy budget allocation:** Two privacy budget allocations within a sliding window are presented by Kellaris et al. (2014). The first approach (Budget Distribution **(BD)**) assigns an exponentially decreasing privacy budget portion to each timestamp. Consequently, the first timestamps in the sliding window consume a higher budget than the last ones. Therefore, this approach favours the earlier timestamps in the window with exponentially more budget than the later ones. The second approach (Budget Absorption **(BA)**) initially allocates an equal privacy budget ratio to all the timestamps within the window. When a private release is unnecessary at a given timestamp, the corresponding privacy budget is reused for later timestamps within the same sliding window. This approach cancels computing a private release at

specific timestamps to avoid exhausting the privacy budget within a sliding window.

- **Last private release approximation:** With a privacy budget allocation strategy comes the necessity to decide the private aggregate to release on the timestamps where the privacy budget is not used. The state-of-the-art approach is to release the last computed private aggregate (Kellaris et al. (2014) and H. Li, Xiong, Jiang, and Liu (2015)).

4.4.3 Experiment Design and Results

We design seven experiments to assess our algorithms' performance. To evaluate the effect of enabling each organization to protect its mobility data stream on utility, experiment 1 captures the added noise variance and experiment 2 varies the noise generation mechanisms. Next, we compare the privacy budget allocation with state-of-the-art methods in experiments 3, 4, and 5. Finally, we assess our strategy to minimize the difference between the accurate mobility aggregate and the private release in experiments 4, 5, 6, and 7. We conduct all the experiments on a Windows 10 machine using Python 3.6.5 with 12GB RAM and AMD Core 4 CPU 2.5GHz. We repeated the experiments 50 times, and below are their descriptions and results:

Experiment 1 (RQ1) Assesses how the overall privacy budget affects the utility of the privately released mobility data stream.

Setup: We fix the sliding window size ($w = 4$), vary the budget ϵ between 1.1 and 1.9 and generate random noise using the Laplace mechanism. We capture the difference between the original mobility aggregate and private release.

Results: As illustrated in Figure 4.5, the medians do not change significantly for different values of ϵ showing that our solution provides a practical utility with a strong privacy guarantee.

Experiment 2 (RQ1) Evaluates the effect of the noise generation mechanism used by each organization on the mobility data stream utility.

Setup: We set the window size to $w = 10$ and vary the total privacy budget ϵ from 1 to 50. Then, we generate the added noise using Laplace and Staircase mechanisms. The value of γ is

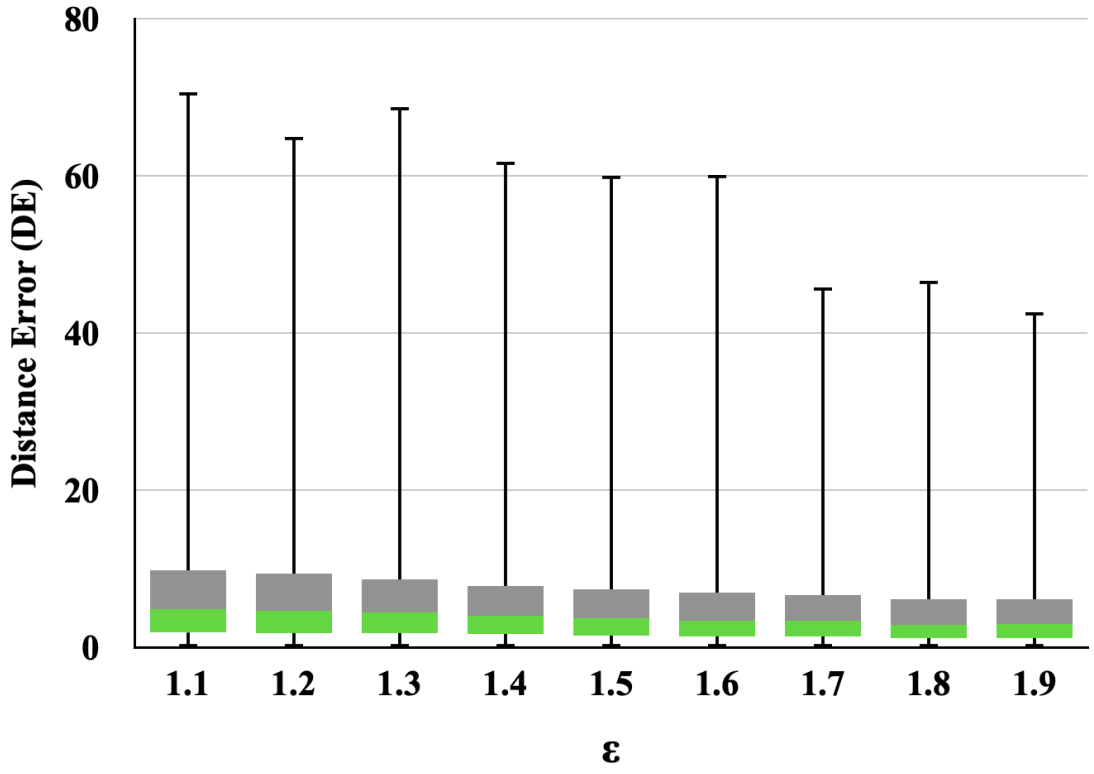


Figure 4.5: Perturbation error vs ϵ

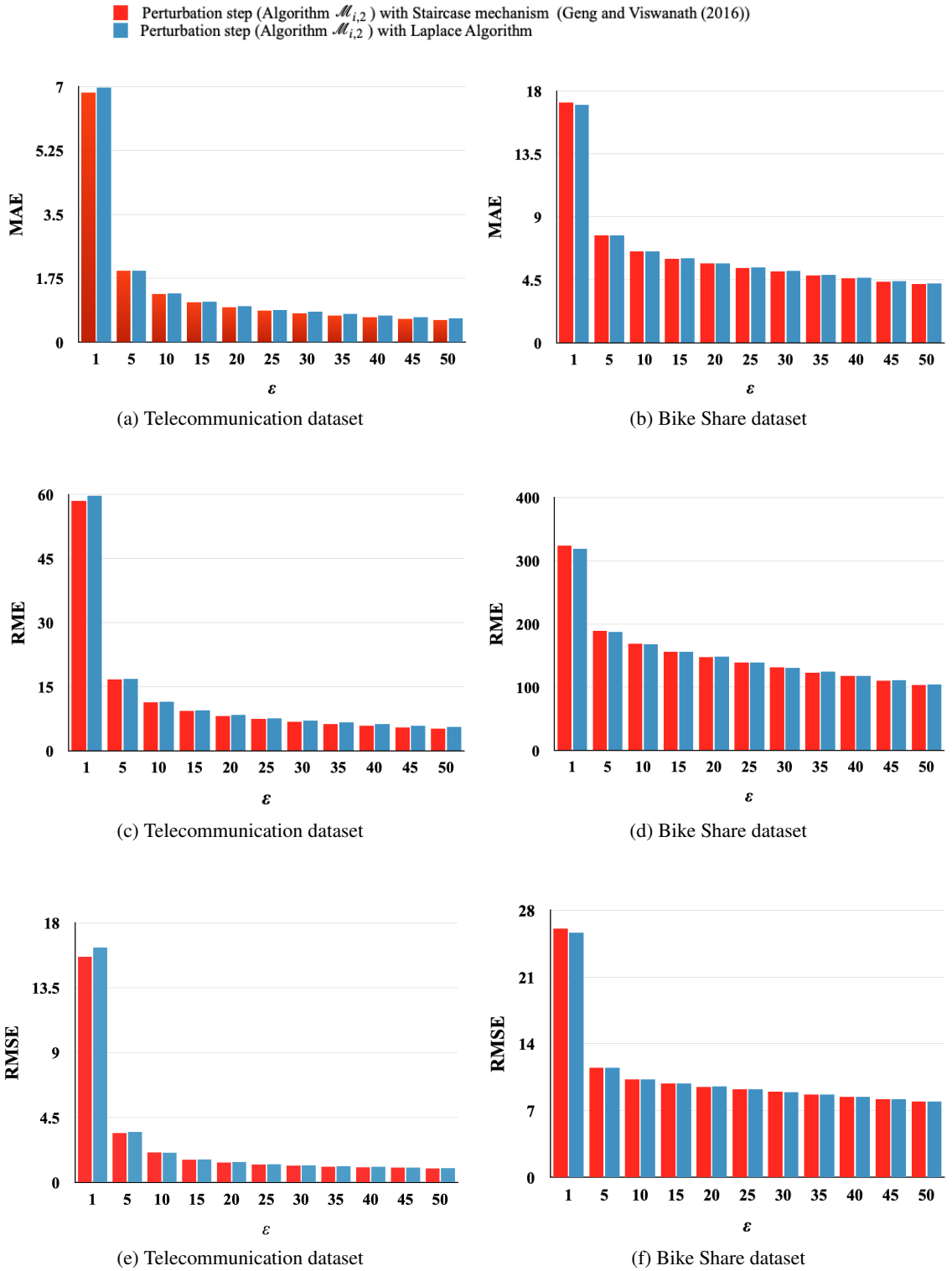
set to $\frac{1}{1+e^{\frac{\epsilon}{2}}}$ for the Staircase algorithm to decrease the amplitude of the applied noise (Geng and Viswanath (2016)). Finally, we calculate the average values of the metrics.

Results: Our proposed privacy budget allocation (Algorithm 3) is not sensitive to the underlying noise generation mechanism. As illustrated in Fig 4.6, the utility of our collective local differential privacy algorithm is similar to either the Laplace or the Staircase mechanisms.

Experiment 3 (RQ2) Compares the utility of the mobility data stream when using Algorithm 3 to the state-of-the-art (Uniform, BA Kellaris et al. (2014), and BD Kellaris et al. (2014)) for different sliding window sizes.

Setup: We vary the window size and fix the privacy budget ϵ to 1.

Results: We observe in Figure 4.7a that the MAE is comparable for our algorithm and BA. Similar to BA, we initialize the privacy budget uniformly. However, unlike BA, we do not cancel future releases because of the unpredictability of the randomized response similarity test. We solve this issue by bounding the privacy budget used by the future timestamps with the remaining privacy



(a) Telecommunication dataset

(b) Bike Share dataset

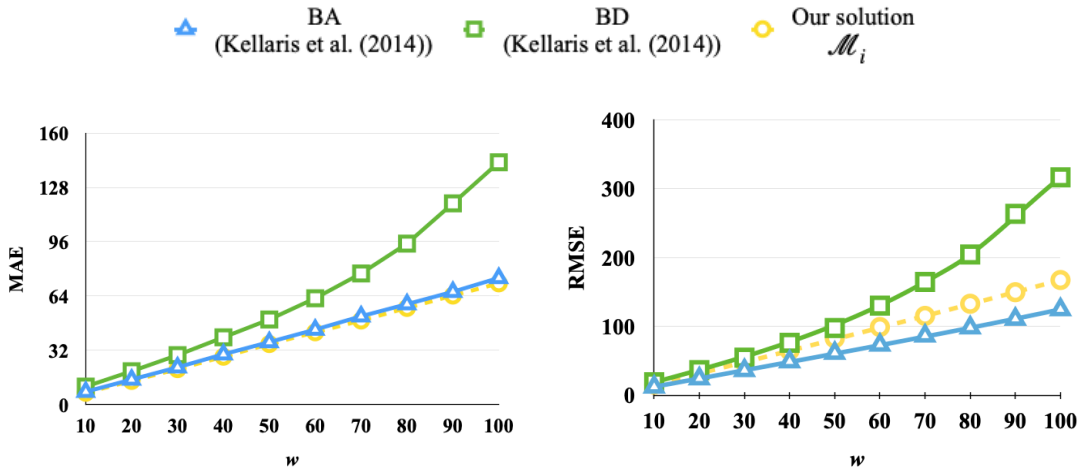
(c) Telecommunication dataset

(d) Bike Share dataset

(e) Telecommunication dataset

(f) Bike Share dataset

Figure 4.6: Effect of the underlying noise generation mechanism on the utility of the collective local differential algorithm



(a) MAE (b) RMSE

Figure 4.7: Perturbation error vs w

budget from the backward projection of the sliding window. The RMSE of our solution is slightly higher than BA, as shown in Figure 4.7b. This difference is because our solution may result in more private releases on a data stream. The total number of releases in a stream in both solutions relies on testing the similarity of the statistics at each timestamp. However, BA cancels some private releases to absorb the privacy budget. In contrast, the total number of releases in our solution depends on the randomized response parameter p that remains mostly the same with the increase of the sliding window size w. BD gives the highest MAE and RMSE compared to BA and our perturbation step.

Experiment 4 (RQ2 and RQ3) Assesses the gain in utility between the last private release (Kellaris et al. (2014)) and the exponential approximation strategies (Algorithm 4) with the perturbation step (Algorithm 3).

Setup: We vary the window size w, set the privacy budget ϵ to 1.0, and capture the average values of the utility metrics.

Results: The exponential step outperforms the last private release (Kellaris et al. (2014)) and the uniform budget distribution based on the three utility metrics. MAE results are shown in Figures 4.8a and 4.8b for the telecommunication, and the Capital Bikeshare data, respectively. We can see the same outcome from Figures 4.8c, 4.8d, 4.8e, and 4.8f for RME and RMSE, respectively. This result further verifies that our exponential approximation strategy reduces the difference between

the true and the private statistics, leading to better utility.

Experiment 5 (RQ2 and RQ3) Evaluates the computational overhead of the exponential approximation strategy (Algorithm 4).

Setup: We vary the sliding window size w, set the privacy budget ϵ to 1.0 and, we calculate the average execution time at each sliding window size with the perturbation step (Algorithm 4) and the state-of-the-art algorithms (BA Kellaris et al. (2014) and BD Kellaris et al. (2014)).

Results: All the presented solutions scale almost linearly with the window size w. The exponential approximation incurs a higher computational cost than the last private release in all the solutions, as illustrated in Figure 4.9. This difference is justified by the additional processing required by the approximation step to sample the closest private release, unlike the last private release (Kellaris et al. (2014)) returns the memorized private statistic from the last release without additional processing.

Experiment 6 (RQ3) Compares the gain in utility between the last private release (Kellaris et al. (2014)) and the approximation step (Algorithm 4) for the state-of-the-art BD (Kellaris et al. (2014)).

Setup: We vary the window size w and set the privacy budget ϵ to 1.0. We execute the experiments 50 times, then calculate the average values of the metrics.

Results: The approximation step improves the utility of BD (Kellaris et al. (2014)). Figures 4.10a and 4.10b show that the approximation step gives smaller MAE values than the last private release (Kellaris et al. (2014)) and the uniform budget distribution. However, RME values of the uniform budget distribution for the bike dataset (Figure 4.10d) are smaller than the last private release and the approximation step. This difference is because BD, with the last private release, gives higher RME values than the uniform budget distribution. Therefore, despite the improvement achieved by the exponential approximation, the resulting RME remains larger than the uniform budget distribution. Moreover, as shown in Equation 16, RME is sensitive to the original value of the data. Therefore, the difference between the accurate and perturbed aggregates is multiplied by the original aggregate. This explains the difference in results between the two datasets. Unlike the bike dataset, the approximation step gives smaller RME values than both the uniform budget distribution and the last private release (Figure 4.10c). For both RMSE (Figures 4.10f and 4.10e)

and RME, the approximation step gives better values than the last private release, improving the utility of the BD algorithm.

Experiment 7 (RQ3) Evaluates the gain in utility between the last private release (Kellaris et al. (2014)) and the approximation step (Algorithm 4) for the state-of-the-art BA (Kellaris et al. (2014)).

Setup: We vary the window size w and set the privacy budget ϵ to 1.0. We execute the experiments 50 times, then calculate the average values of the metrics.

Results: The approximation step improves the utility of BA (Kellaris et al. (2014)). MAE results are illustrated in Figures 4.11a and 4.11b for the telecommunication and the Capital Bikeshare data, respectively. Similar to Experiment 7, the approximation step outperforms the last private release and the uniform budget distribution. The same outcome is illustrated for RME (Figures 4.11c and 4.11d) and RMSE (Figures 4.11e and 4.11f). This result shows that our approximation step leads to a better utility.

4.4.4 Discussion

Our experimental evaluation showcased the utility improvement of our budget allocation and approximation method. For instance, the medians of the added noise do not change significantly for different values of ϵ shows the practicality of our local privacy mechanism. Moreover, our solution supports multiple noise-generating mechanisms (Laplace or Staircase probability distribution). Finally, based on the comparison experiments, it is clear that our solution outperforms state-of-the-art methods in absolute and relative error.

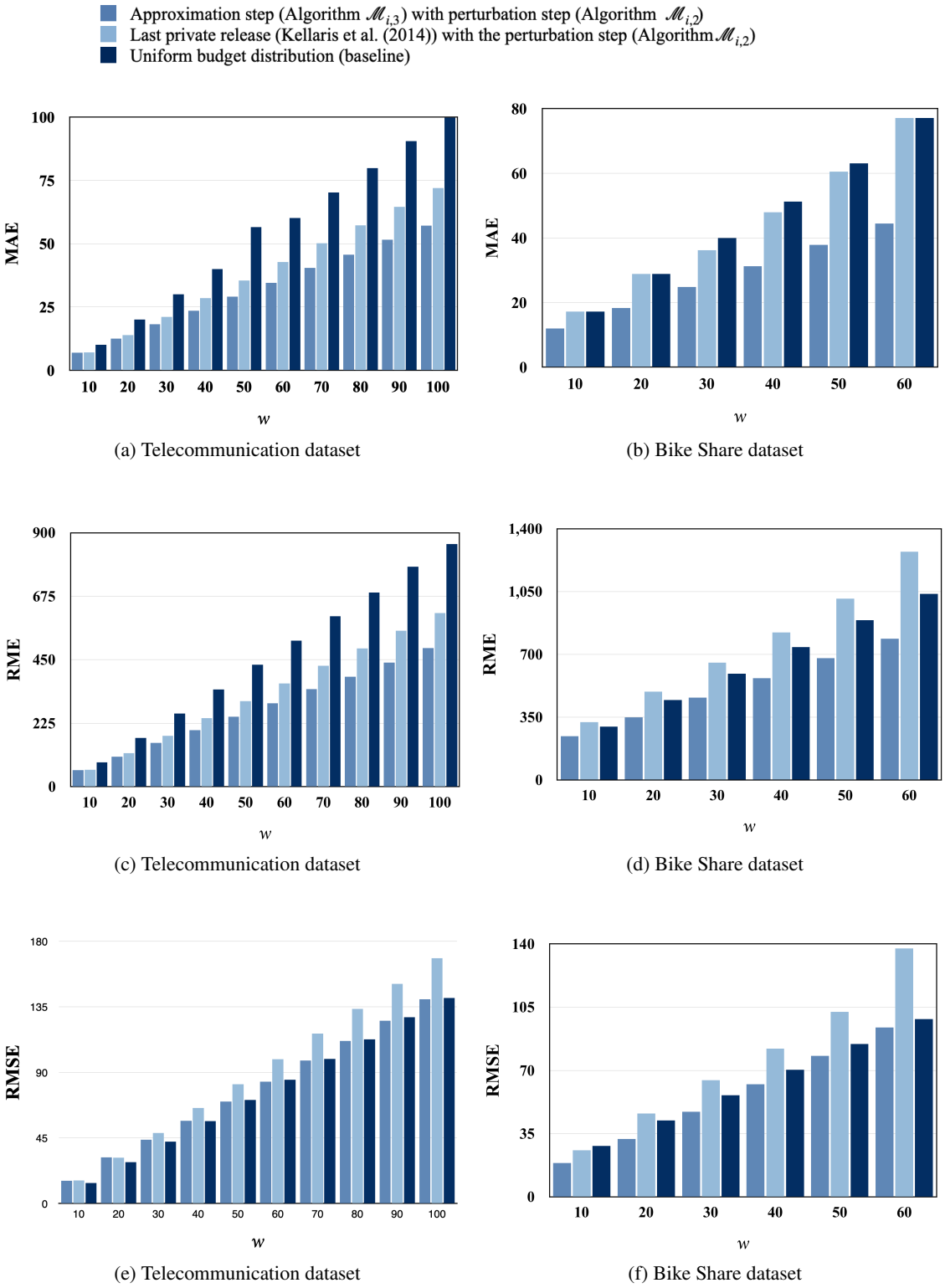
4.4.5 Threat to Validity

We identify one source of threat to validity. The state-of-the-art algorithms (Kellaris et al. (2014)), as well as our algorithm, are data-dependent. The decision step result relies on the similarity between the current statistic and the last private release. Consequently, we chose two different real-life datasets for our utility assessment. The telecommunication dataset is captured from a real-time environment for only one hour, while the Bikeshare dataset is historical data of four months. Also, the Bikeshare dataset contains 368 data streams with different lengths. This difference gives

the advantage of testing the utility of the approximation strategy in datasets of varied natures, proving our solution's validity.

4.5 Summary

In this chapter, we design an algorithm guaranteeing local differential privacy for location statistics collective release. We design a dynamic privacy budget allocation and define a new approximation strategy that reduces the perturbation error. We leverage the exponential algorithm to release the private and closest statistic to the accurate location statistic. Also, we formally analyze our method's privacy and utility. Experiments on real datasets demonstrate that our algorithm enhances the existing state-of-the-art techniques' utility. In the future, we want to improve the performance of the exponential approximation strategy execution time. Finally, we would like to investigate answering more sophisticated queries on the collective data, such as range queries, with little utility loss.



(a) Telecommunication dataset

(b) Bike Share dataset

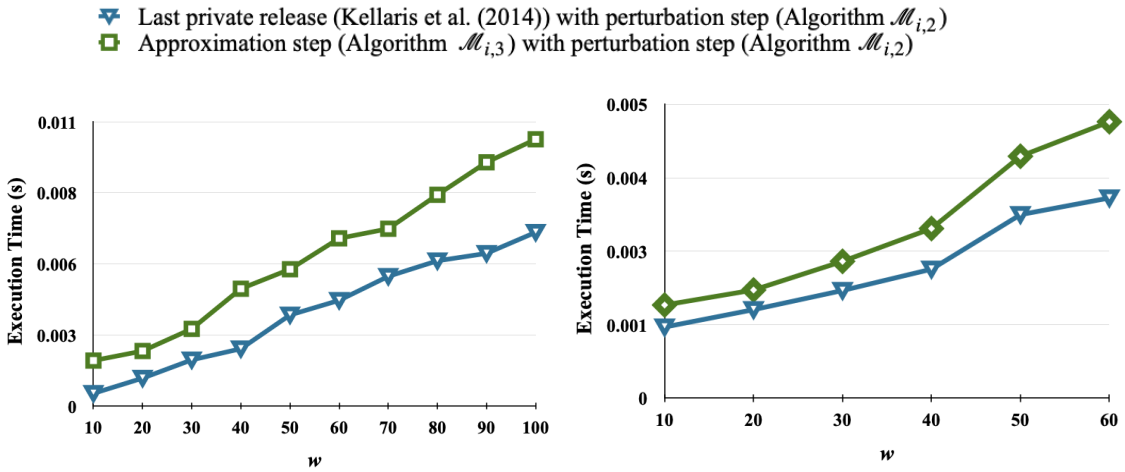
(c) Telecommunication dataset

(d) Bike Share dataset

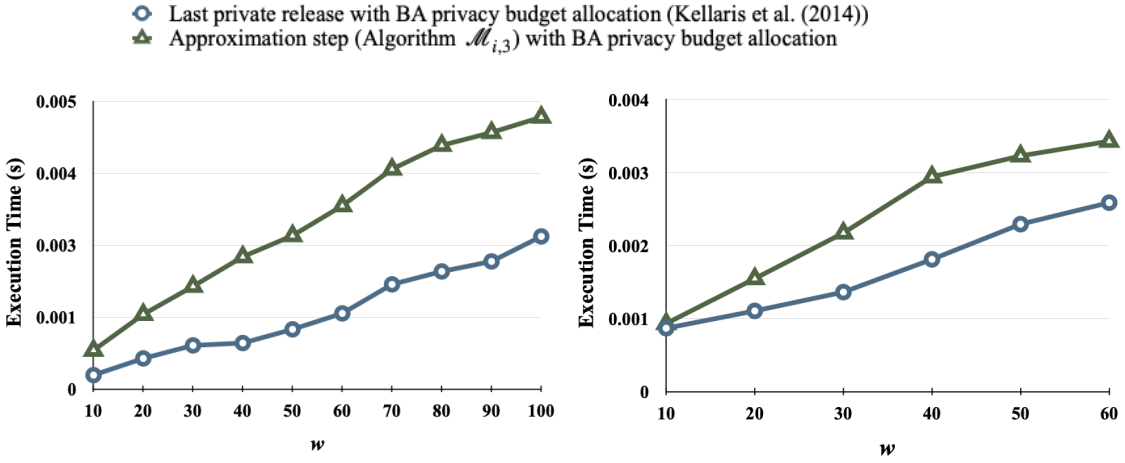
(e) Telecommunication dataset

(f) Bike Share dataset

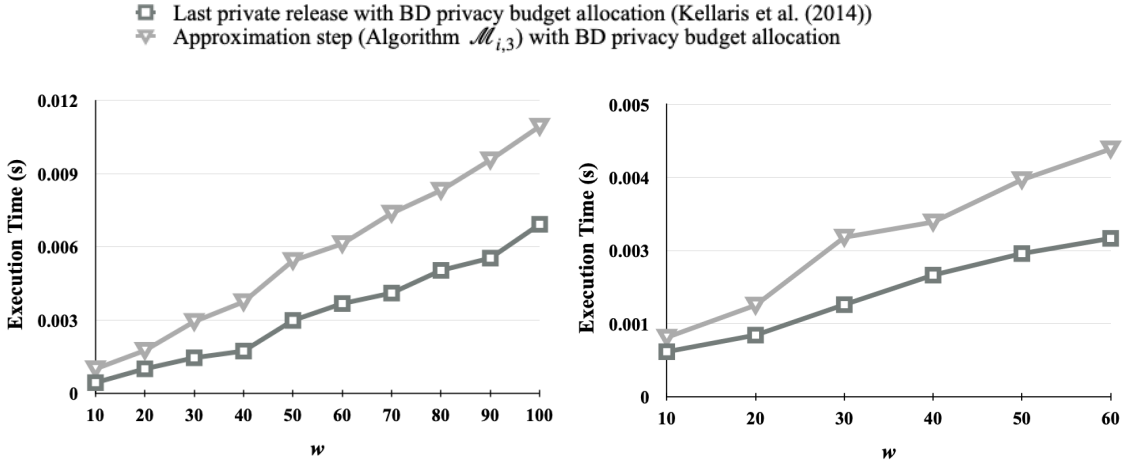
Figure 4.8: Effect of the approximation strategy on the collective local differential privacy algorithm utility



(a) Telecommunication dataset (b) Bike Share dataset

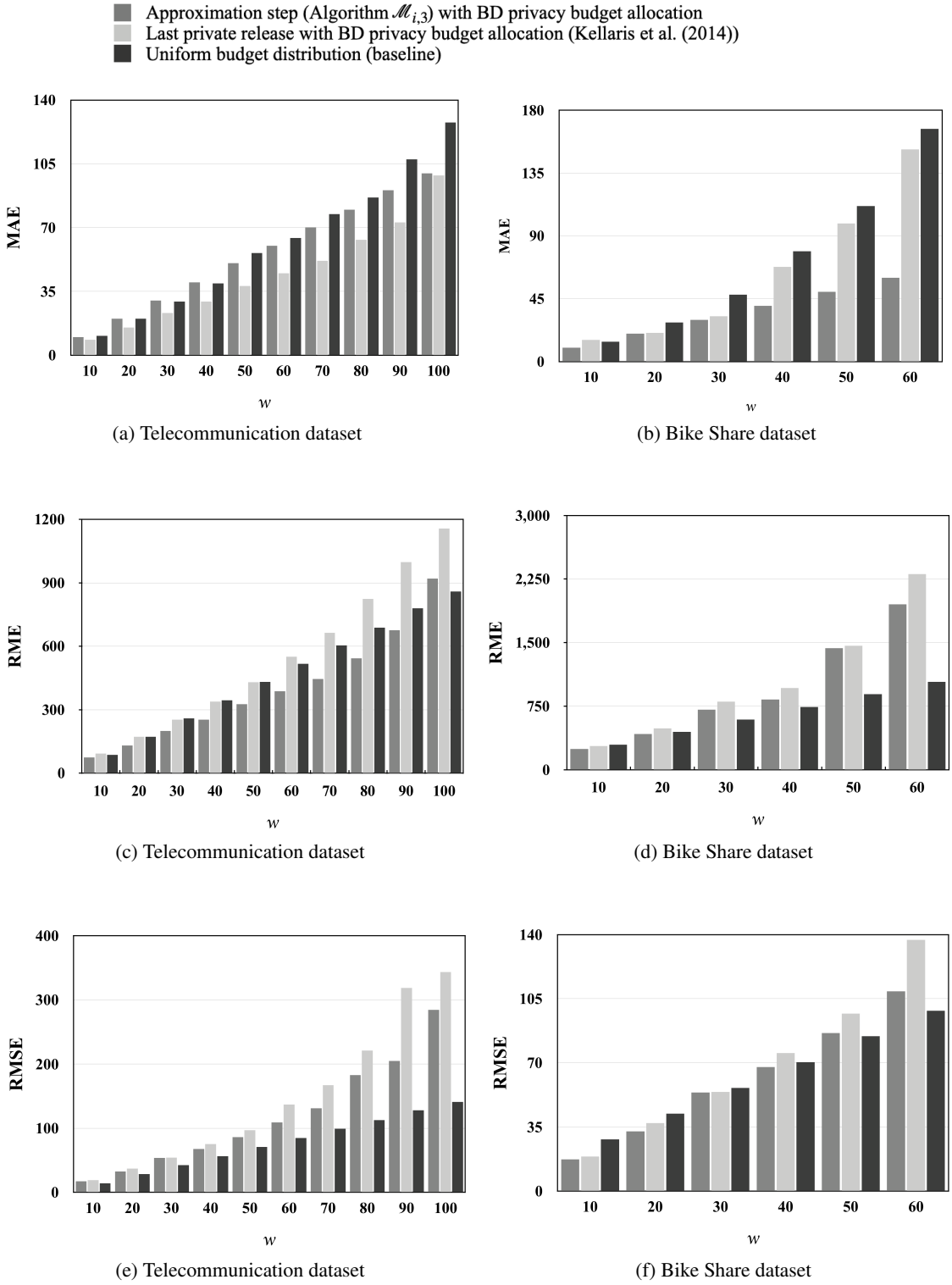


(c) Telecommunication dataset (d) Bike Share dataset



(e) Telecommunication dataset (f) Bike Share dataset

Figure 4.9: Effect of the window size w on the execution time of the collective local differential private algorithm (4.9a, 4.9b), the budget absorption approach (4.9c, 4.9d), and the budget distribution approach (4.9e, 4.9f)



(a) Telecommunication dataset

(b) Bike Share dataset

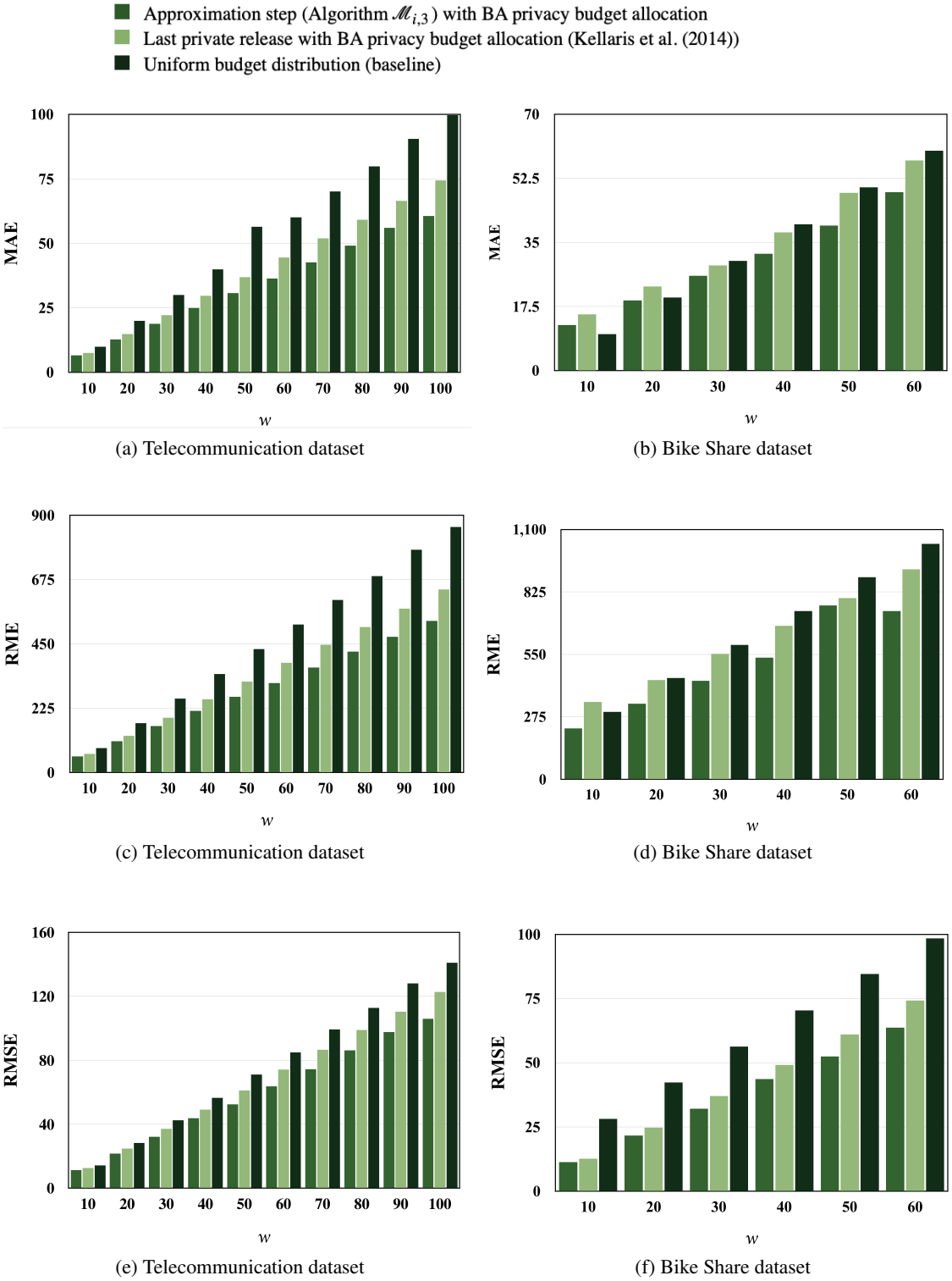
(c) Telecommunication dataset

(d) Bike Share dataset

(e) Telecommunication dataset

(f) Bike Share dataset

Figure 4.10: Effect of the approximation strategy on the utility of budget distribution approach



(a) Telecommunication dataset

(b) Bike Share dataset

(c) Telecommunication dataset

(d) Bike Share dataset

(e) Telecommunication dataset

(f) Bike Share dataset

Figure 4.11: Effect of the approximation strategy on the utility of the budget absorption approach

Chapter 5

A Mobility Forecasting Framework with Vertical Federated Learning

5.1 Introduction

With the prevalence of mobile devices and location-based services, forecasting human mobility has become essential in ubiquitous computing. For example, mobility data may help alleviate traffic congestion (Kothai et al. (2021)), control traffic lights (H. Wei, Zheng, Yao, and Li (2018)), and improve the efficiency of traffic operations (Rao and Vazquez (2020)). Moreover, data on travel history can help evaluate the effectiveness of non-pharmaceutical interventions and forecast the spread of COVID-19 (Ilin et al. (2021)).

Currently, most mobility forecasting approaches (Singh et al. (2020), W. Li et al. (2019), and S. Guo, Lin, Li, Chen, and Wan (2019)) adopt a centralized learning strategy by transferring the entire mobility data to a data center to learn the prediction model. Alternatively, distributed learning methods (Feng, Rong, Sun, Guo, and Li (2020) and Zhang et al. (2021)) focus on horizontal federated learning, where each organization has a subset of samples covering the entire mobility location domain. However, mobility data typically pertains to independent organizations with different locations regions. Therefore, the learning methods need to cater to vertically distributed data.

The first challenge we identify in mobility vertical federated forecasting is that since each organization only holds a location domain subset, none can tackle a forecasting model encompassing

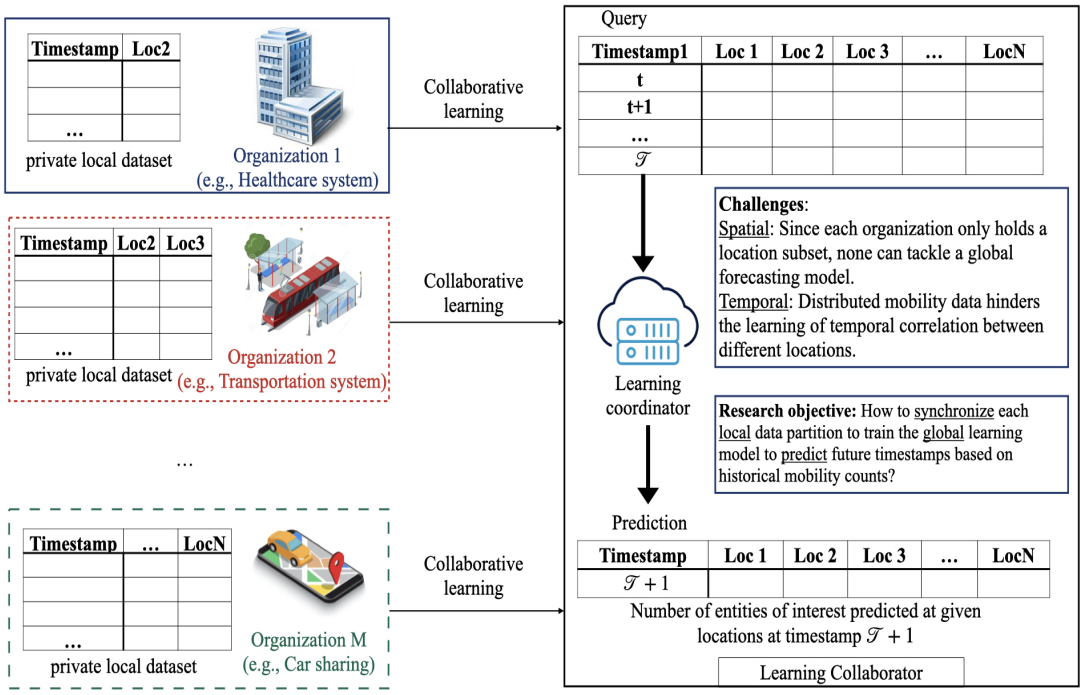


Figure 5.1: System model: our main focus is to design a learning framework that coordinates the learning between multiple organizations to make mobility forecasting.

the whole location domain. Therefore, this raises the following research question:

> RQ1: How to achieve forecasting on the entire location domain when the individual locations belong in separate organizations' datasets?

Vertically distributed mobility data compromises the spatio-temporal correlation between locations hindering learning. Hence, compromising the forecasting accuracy. Therefore, we define the second research question as follows:

> RQ2: How can the spatio-temporal correlation between distributed locations be reconstructed to provide mobility count forecasting?

This work proposes a mobility vertical federated forecasting (MVFF) framework that allows the learning process to be jointly conducted over vertically partitioned data belonging to multiple organizations. MVFF enables the forecasting of mobility predictions covering a joint location domain. We summarize our contributions as follows:

- Mobility Vertical Federated Forecasting (MVFF) framework is a cross-organization collaborative learning framework composed of three components: local models, a global model, and a learning collaborator. The local models independently embed the spatio-temporal correlation in each organization dataset. The global model combines the learning from the local models to incorporate the correlation between all locations in the location domain. The learning collaborator synchronizes the learning between the global and local models.

- Temporal synchronization module (RQ1): resides in the learning collaborator. Leverages a buffer that stores the local model parameters sent from the organizations with their corresponding timestamps. The timestamps correspond to the training period time performed by the organization.

- Spatial synchronization module (RQ2): resides in the learning collaborator. Leverages a map building that organizes the local model parameters to cover the location domain, then triggers the spatial correlation learning via the global model.

- We evaluate MVFF's performance over two real-world datasets with four variables: two local (LSTM, GRU) and two global (CNN, GNN) models. We also perform a comparison with other state-of-the-art models. Experimental results demonstrate that the two datasets' mean percentage error performance gains are up to 12% and 4% compared to the state-of-the-art, respectively.

The rest of this chapter is structured as follows. Section 5.2 gives the details of the problem definition. Section 5.3 describes the details of the framework, and Section 5.4 explains the experimental evaluation results.

5.2 Problem Statement

In this section, we introduce the system model in which we propose our solution and present the problem's definition.

5.2.1 System Model

We consider a set of M organizations (or data owners) $\mathcal{O} = \{O_1, ..., O_M\}$ that jointly train a learning model without sharing their respective datasets, as illustrated in Figure 5.1. The learning coordinator, $\mathcal{C}$, is responsible for orchestrating the global learning objective across the organizations.

We denote $D_i = \{x_t^{D_i}\}_{t=1}^{d}$ where $x_t^{D_i}$ represents the t-th row of the organization O_i, and each column corresponds to a location. Each row, $x_{t,L_j}^{D_i}$, corresponds to the count of entities of interest at location L_j. Finally, n is the combined number of locations from all organizations.

After obtaining the trained learning model, $\mathcal{C}$ makes predictions $X_{\mathcal{T}+1} = \{x_{\mathcal{T}+1}^{L_1}, x_{\mathcal{T}+1}^{L_2}, ..., x_{\mathcal{T}+1}^{L_n}\}$ denoting the mobility counts at locations $\{L_1, ..., L_n\}$, at timestamp $\mathcal{T}+1$. Table 5.1 summarizes the frequently used notations.

Notation	Description
O_i	Organization i
D_i	Organization i's dataset
M	Number of organizations
$\mathcal{C}$	Learning coordinator
$x_t^{D_i}$	the t-th row of the organization O_i
n	Number of locations
$\mathcal{T}$	Mobility count observations length
$X_{\mathcal{T}+1}$	Predictions of mobility counts at timestamp $\mathcal{T}+1$
$\mathcal{I}$	Training interval length
M_i	Local learning model of organization O_i
G	Global learning model
ϕ	Location domain map

Table 5.1: Used notations for mobility forecasting with vertical federated learning

5.2.2 Problem Formulation

To predict the mobility counts at timestamp $\mathcal{T}+1$, the learning coordinator $\mathcal{C}$ must synchronize the global learning between the organizations. In particular, the learning coordinator must maintain a correspondence between the mobility data in the consecutive timestamps at each organization. Moreover, the learning coordinator must model the correlation between all the locations in the location domain. Thus, we define the problem as spatio-temporal coordination of the learning from the organizations $\mathcal{O}$ to predict the mobility count predictions $X_{\mathcal{T}+1}$ given historical mobility count

observations of $\mathcal{T}$ timestamps $\{X_1, X_2, ..., X_{\mathcal{T}}\}$.

5.3 Mobility Vertical Federated Forecasting

This section provides the details of our proposed MVFF framework. First, we present the overall framework. Then, we describe the temporal synchronization and the temporal synchronization modules.

5.3.1 Overall Learning Framework

We propose to perform vertical federated learning where the collaborator combines the parameters received from the organizations' models to train a global model that can forecast the location domain's mobility counts. The framework comprises three components: the local models, the global model, and the learning collaborator. Figure 5.3 illustrates the components and their interactions during the training process.

We consider that each organization, O_i, has a local model M_i that uses the organization's data for training. The M organizations support the learning collaborator to train a global model encompassing the location domain, ϕ. We assume that the local and global models do not share the same neural network architecture. This difference is a consequence of the vertical partitioning of the mobility dataset. Consequently, the learning objective of the local model is to conceptualize the spatio-temporal correlations between locations under the organization's jurisdiction only. In contrast, the global model's neural architecture covers the entire location domain. The temporal and spatial synchronization modules bridge the connection between local and global models.

Algorithm 5 gives the high-level steps of MVFF. The learning collaborator and organizations initialize all their models' parameters before the learning starts. We assume that the organizations communicate their local model's architecture and location topology to the learning collaborator at the initialization of the training. Each organization, O_i, selects a timestamp t and builds a mobility data batch, B_i, using samples with timestamps between t and $t + \mathcal{I}$ (line 3). These samples consist of sequences of consecutive mobility data with the length of the historical data $\mathcal{T}$ and the corresponding target counts, as illustrated in Figure 5.2. O_i then performs a training iteration of

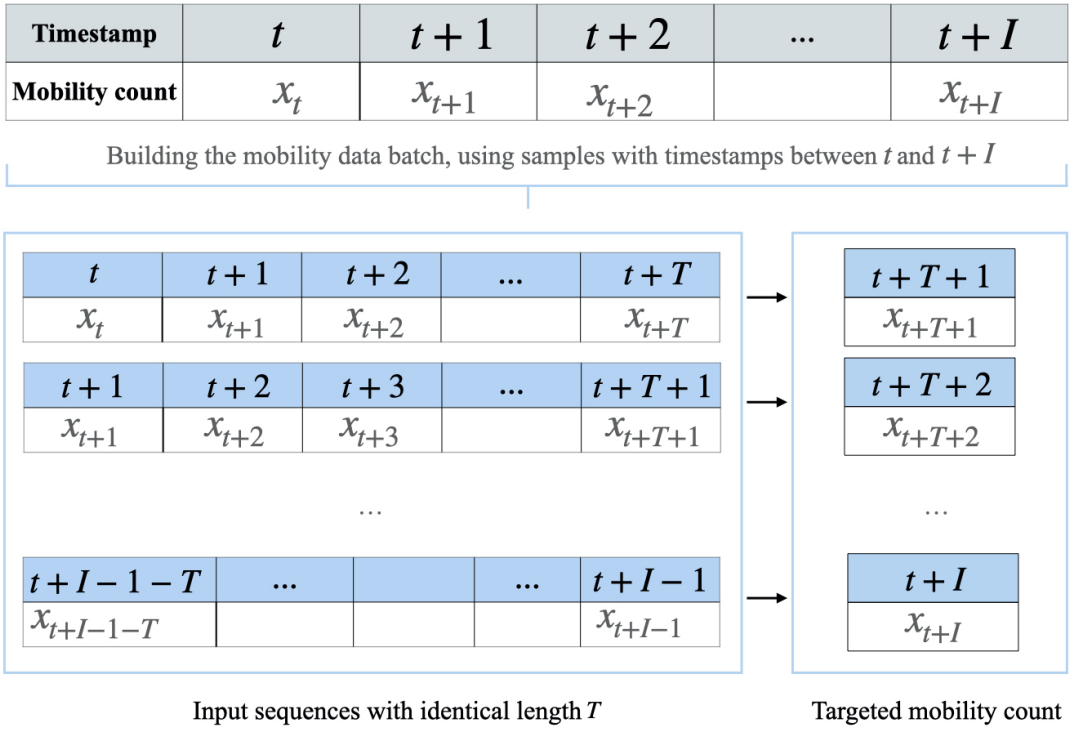


Figure 5.2: Building the training batch: a sequence of $\mathcal{I}$ mobility counts with consecutive timestamps is restructured into sequences with the historical data length $\mathcal{T}$. The targeted mobility count represents the ground truth prediction.

the local model with the built batch (line 4). Next, O_i sends the resulting model parameters with the corresponding timestamp, t, to the learning collaborator (line 5). Upon receiving the learning parameters, the learning collaborator uses them to build a training batch for the global model (lines 8-14). Finally, the learning collaborator computes the global model parameters and updates them (lines 15-16). This collaboration continues until the predefined number of iterations T is satisfied.

It is essential to notice that the learning collaborator $\mathcal{C}$ communicates the interval length $\mathcal{I}$ to the organizations at the initialization of the learning process. Moreover, when the collaborator finishes the global model training, it broadcasts the *Halt* command to all organizations. Meanwhile, the organizations continuously send their local model updates to the learning collaborator until they receive a *Halt* signal. Finally, the organizations can build their training batches with different starting timestamps. Hence, the learning collaborator is responsible for synchronizing the multiple parameters. We next describe the detailed procedures of the temporal and spatial synchronization modules, respectively.

Algorithm 5 Mobility vertical federated learning

Require: Mobility count observations length $\mathcal{T}$, temporal buffer size $\mathcal{I}$, location domain map ϕ, number of iterations T **Initialize:** M_i parameters for every organization, global model parameters

//Learning process at i-th organization

while not *Halt* **do**
  for timestamp t in D_i **do**
    Build a data batch B_t with mobility counts in the interval $[t, \mathcal{I}]$
    Compute gradient $\Delta\theta_i$ of local model M_i on batch B_t
    Send (t,$\Delta\theta_i$)) to learning collaborator $\mathcal{C}$
  end for
end while

//Learning process at collaborator $\mathcal{C}$

for each iteration j from $1, 2, ..., T$ **do**
  When $\mathcal{C}$ receives $\Delta\theta_i$ from O_i, the temporal synchronization module stores it to a buffer dedicated to the local model M_i
  The temporal synchronization module sorts the buffers and samples a timestamp t
  for each location L_i in Φ **do**
    The spatial synchronization module builds a mobility count batch B based on the corresponding local model's parameter $\Delta\theta_t$
  end for
  The spatial synchronization module organizes the batch B following the location domain map structure ϕ
  Compute gradient $\Delta\theta_c$ of global model G on batch B
  Update global model G's parameter θ_c
end for
Send *Halt* command to every organization

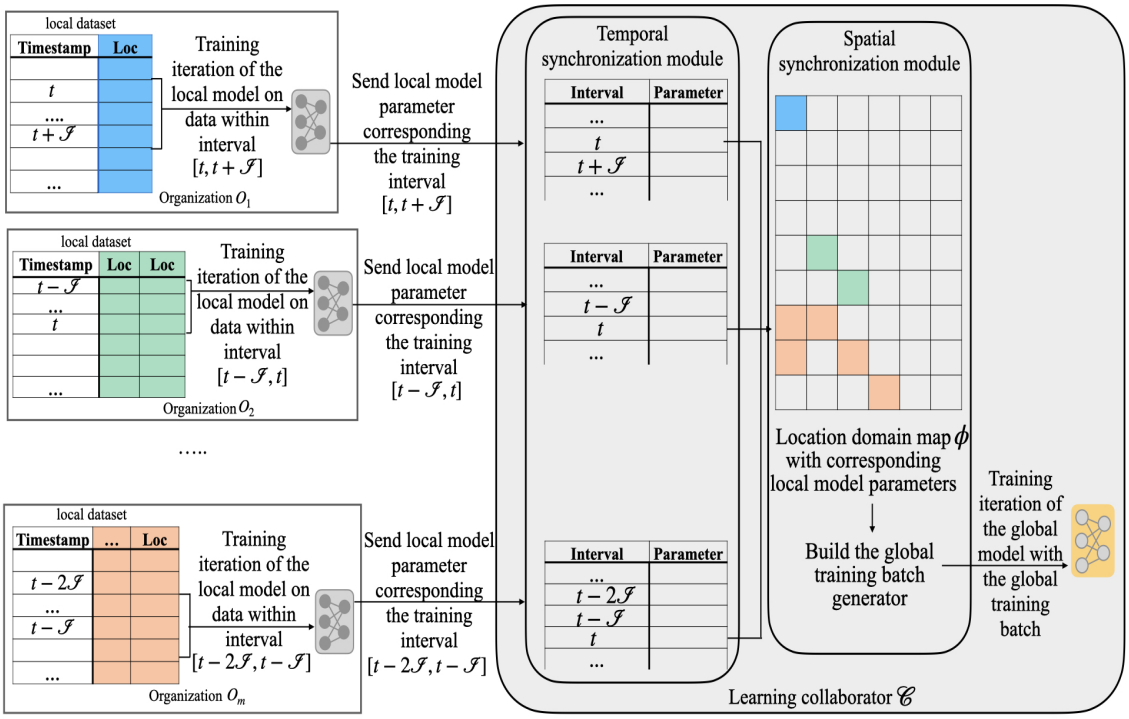


Figure 5.3: Illustration of the training procedure in the MVFF framework. The global model trains mobility data generated using the temporally synchronized local models' training parameters and the location domain map. The different colours on the location map indicate different location features (mobility count) coming from the corresponding organizations.

5.3.2 Temporal Synchronization Module

The temporal discrepancy between mobility data is a direct consequence of the vertical partitioning of mobility data. Hence, the goal of the temporal synchronization module is to correlate between the organizations' learned parameters at the same timestamp. To achieve this, we utilize ordered buffers at the temporal synchronization module.

The temporal synchronization module creates one parameter buffer per organization. The buffers are initialized at the beginning of the training procedure. When the temporal synchronization module receives a model parameter and timestamp from an organization, it stores them in the dedicated buffer in ascending order of the timestamp value. When all the buffers have parameters corresponding to any given timestamp t, the temporal synchronization module sends them to the spatial synchronization module.

5.3.3 Spatial Synchronization Module

Another consequence of the vertical partitioning of the mobility data is the loss of correlation between locations. For example, two spatially adjacent locations may belong to two different organizations. The mobility counts in the two locations may be mutually interdependent, such as in the case of crowd counts. However, it is challenging to assess this interdependence when the mobility counts of the two locations pertain to separate datasets. The goal of the spatial synchronization module is to reconstruct the correlation between locations through the organizations' learned parameters.

The spatial synchronization module utilizes the location domain map ϕ and the local model parameters to generate a training batch for the global model. We assume that the synchronization module has a training mobility count dataset. Depending on the application domain, this dataset can be minimal and easy to obtain in practice, such as the publicly available datasets (Geolife Zheng, Xie, Ma, et al. (2010), T-Drive Yuan et al. (2010), or Gowalla E. Cho, Myers, and Leskovec (2011)).

Similar to the training batches built for training the organization's local model, the global training batch builds sequences with the historical data length $\mathcal{T}$ using the global training mobility count dataset. However, the targetted mobility counts corresponding to each sequence are generated using the local models' parameters. Since the spatial synchronization module has copies of the organization's models' neural architectures, it uses the model parameters received from the temporal synchronization module to compute the targeted mobility counts. As illustrated in Figure 5.3, the synchronization module coordinates each local model on the location domain map to the corresponding location. Then, it uses the local model to predict the targeted mobility counts of each sequence in the global training batch. Next, the spatial synchronization module uses the predictions as ground truth to train the global model. Finally, the spatial synchronization module triggers a global model training iteration over the global training batch.

5.4 Experimental Evaluation

In this section, we conduct extensive experimental analyses to validate the efficiency and utility of MVFF.

5.4.1 Datasets and Metrics

We evaluate our proposed model using two large-scale real-world datasets from New York City (Gordon-Koven and Levenson (2014)) and Yelp (Yelp (2019)). Each dataset contains mobility data as follows:

New York City Bike dataset (Gordon-Koven and Levenson (2014)): It contains monthly trip details of bikes based on the bike stations deployed in New York City as described below:

{*TripId, Trip Duration, Start Time, Stop Time, Start Station ID, Start Station Name, Start Station Latitude, Start Station Longitude, End Station ID, End Station Name, End Station Latitude, End Station Longitude*}

We divide the city into 16x8 regions, each containing bike stations based on their latitude and longitude. We use the start and end stations and the start and stop times to generate a sequence of bike counts at each region, with a time granularity of 15 minutes. We collect the first four months of 2021: February 1st to Mai 31st, 2021. Therefore, the total number of timestamps in the dataset is 1.5 million.

Yelp review dataset (Yelp (2019)): It contains 192k businesses from metropolitan areas in the United States with their location (latitude and longitude), and 6.68M timestamped users reviews as below:

{*review id, user id, business id, review, date*}

We split the country into a uniform grid of 88 regions, each containing businesses based on their latitude and longitude. We use the reviews' timestamps to generate a sequence of review counts at each region, with a time granularity of 6 hours. The review dates start on January 1st, 2018 and stop on December 31st, 2020. Overall, the sequence dataset contains 280k timestamps.

We select two metrics for utility assessment, root mean square error (RMSE) and weighted mean absolute percentage error (WMAPE). We compute the two metrics as follows: let x_t be actual

mobility count at any time instance t and x'_t be the corresponding predicted count, where t varies from 1 to n, consequently:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(x_t - x'_t)^2} \tag{18}$$

$$WMAPE = \frac{\sum_{t=1}^{n}|x_t - x'_t|}{\sum_{t=1}^{n}|x_t|} \tag{19}$$

WMAPE is less sensitive to outliers than RMSE since WMAPE focuses on absolute errors ($|x_t - x'_t|$). However, both metrics do not include the sign of the prediction error. Therefore, adding the average error (AE) metric is necessary to see if the forecasting techniques underestimate or overestimate the mobility count. The average error (AE) will be reported for each case as follows:

$$AE = \frac{1}{n}\sum_{t=1}^{n}(x_t - x'_t) \tag{20}$$

Dataset	Min	Max	Mean	Median	Standard deviation
New York City Bike	0	50.70	3.96	6.24	6.94
Yelp review	0	6.73	0.41	0.72	0.89

Table 5.2: Basic analysis of the used datasets

5.4.2 Framework Implementation with Model Variations

We study the accuracy of our vertical federated learning framework using different learning models. Since the framework consists of local and global models, we implement them as different neural networks that are commonly utilized for mobility forecasting to form the following variations:

- **GRU+CNN**: The local models in this variation implement the GRU architecture with one unit, followed by a linear layer that maps the features to predictions. We interpret the location grids as images where longitude and latitude are equivalent to width and height, and timestamps perform as channels. Therefore, the global model has CNN architecture that comprises

a three-dimensional convolutional layer to reveal the activity patterns between timestamps. Stride and padding for convolutional layers were selected to preserve the original size of the grid.

- **LSTM+CNN**: The local models in this variation implement the LSTM architecture with one LSTM cell followed by a linear layer that maps the features to predictions. LSTM takes as input the times series with the number of people in each region. The global model follows the same architecture as the CNN in the GRU+CNN variation.

- **GRU+GNN**: In this variation, the local model follows the same architecture as the GRU in the GRU+CNN variation. The global model implements the Graph Attention Networks architecture (Veličković et al. (2018)), utilizing the attention mechanism to obtain the correlations between locations in the grid. We implement an attention layer for each cell that factors the spatial and temporal mobility data from the cell's neighbours on the grid. The adjacency matrix encodes the neighbourhood relationship between the cells in the location grid. We compute the attention score for any pair of nodes in the graph. Then, we use a LeakyReLU (Leaky Rectified Linear Units) and softmax activation functions to normalize the attention score and obtain the attention coefficient. Finally, we filter the obtained attention coefficient to retain the attention coefficients only for connected node pair following the adjacency matrix.

- **LSTM+GNN**: In this variation, the local model follows the same architecture as the LSTM+CNN variation, while the global model follows the same architecture as the GRU+GNN variation. Following the guidelines detailed by Greff, Srivastava, Koutník, Steunebrink, and Schmidhuber (2016), we implement a single LSTM cell followed by a linear layer that maps the features to predictions. Adding further layers complicates the training without giving better results.

5.4.3 Comparison Frameworks

We compare our model with the following federated frameworks for time-series forecasting to investigate utility improvement:

- **FedGRU Y. Liu, Zhang, et al. (2020)**: This framework uses the FederatedAveraging (McMahan, Moore, Ramage, Hampson, and y Arcas (2017)) algorithm to build a global GRU model based on model parameters from different organizations. Each organization uses its local data to perform a gradient descent optimization on its local model. Then the central cloud performs a weighted average aggregation of the local model updates uploaded by the clients. Consequently, the framework uses the same network architecture in the global and local models.

- **Federated LSTM Taïk and Cherkaoui (2020)**: This framework uses LSTM on edge devices that capture houses' electrical loads to predict future ones. The studied group of houses has similar geographical and infrastructural properties. The edges train the local LSTM models using a dataset composed of sliding windows with predetermined look-back steps. The initial global LSTM model is pre-trained using publicly available data or initialized randomly. Then, the cloud server aggregates the local model updates to compute the global model using the FederatedAveraging algorithm (McMahan et al. (2017)).

5.4.4 Experiment design and results

We set the MVFF parameters described in Algorithm 5 as follows. We fix the mobility count observations length $\mathcal{T}$ to 6 and forecast 1 step ahead. The temporal buffer size $\mathcal{I}$ for Experiments 1 and 2 is set as 24 hours, and we vary this parameter in Experiment 3. The location domain map ϕ for each dataset corresponds to a grid of 16x8 and 8x8 for New York City Bike and Yelp review, respectively. We consider the highest possible vertical distribution. Hence, the number of organizations corresponds to the number of cells in the grid of each dataset.

The training dataset for each organization's local model comprises the mobility counts of the corresponding grid cell. We build the training data for the global model by randomly sampling 10% of the local datasets and testing data with 10% of each organization's local dataset.

We implement MVFF and the comparison frameworks using Pytorch (Paszke et al. (2019)) and Colab platform. We use Adam (Kingma and Ba (2014)) with a learning rate of 0.001 for the local and global models. Our code is available at https://github.com/FatimaErrounda/MVFF.

Experiment 1 (RQ1) Assesses the utility of the variations with the comparison frameworks.

Configuration: Report the utility metrics (Equations 1, 2, and 3) based on the variations' global models' forecasting over the testing data. Capture the utility metrics for the comparison frameworks using the same testing data. For simplicity, we average the metric values over all the cells in the dataset grid and round the values to two decimal places.

Results: Table 5.3 and Table 5.4 show the results for the New York City Bike and Yelp review datasets, respectively. The four variations deliver smaller metric values than the comparison frameworks. This improvement is due to the inferred location correlation achieved by the spatial synchronization module. Moreover, we notice that RMSE and AE values for Yelp are smaller than the New York City Bike. This difference can be attributed to the variability of the mobility counts per dataset. We calculate the mean and standard deviations for the New York City Bike and Yelp review datasets. Table 5.2 supports our analysis; as illustrated, the variability of the New York City Bike dataset is higher than Yelp. Moreover, Figure 5.4 shows a portion of the two test datasets to better illustrate the variability in the mobility counts. GNN-based variations give the smallest metric values.

Model	RMSE	WMAPE	AE
LSTM+CNN	12.23	1.07	7.81
GRU+CNN	15.62	0.99	7.86
LSTM+GNN	6.88	0.58	2.08
GRU+GNN	6.79	0.58	1.93
FedGRU Y. Liu, Zhang, et al. (2020)	17.14	1.00	8.90
Federated LSTM Taïk and Cherkaoui (2020)	17.24	1.18	8.95

Table 5.3: The metrics results on Bike New York dataset

	RMSE	WMAPE	AE
LSTM+CNN	1.10	1.14	0.63
GRU+CNN	1.07	1.17	0.76
LSTM+GNN	0.95	1.11	0.18
GRU+GNN	0.96	1.12	0.19
FedGRU Y. Liu, Zhang, et al. (2020)	1.22	1.51	0.66
Federated LSTM Taïk and Cherkaoui (2020)	1.24	1.14	0.68

Table 5.4: The metrics results on Yelp dataset

Experiment 2 (RQ1) Assesses the framework's forecasting compared to the ground truth mobility data.

Configuration: Since the location domain includes multiple locations, we select one cell per dataset to display the ground truth and the predictions made by the four variations and the comparison frameworks.

Results: As shown in Figure 5.5, the four variations forecast closely follow the trend of the oscillations of the ground truth, with the GNN-based variations as the nearest. The predicted counts produced by the comparison frameworks noticeably exceed the ground truth at some timestamps. None of the models could predict the sharp peaks in the New York Bike dataset. However, the GNN-based variations can predict the peaks with fewer counts than occurred. This improvement is due to better capturing of the spatial features of mobility counts of the GNN model.

Experiment 3 (RQ2) Assesses the impact of the interval length parameter in reconstructing the spatio-temporal correlation between locations.

Configuration: Vary the temporal buffer size $\mathcal{I}$ and capture the utility metrics for the 4 variations.

Results: Figure 5.6 demonstrates that the framework's performance is orthogonal to the temporal buffer size. This result is the consequence of the sequential aspect of creating the local training batch for each organization. As explained in Section 5.3, training the local models on a given interval leads to the generations of sequences with the length of the historical mobility data. Therefore, a more extended temporal buffer size means that generated sequences number increase. Nevertheless, the correlation between locations is encoded in the sequences' counts, not their numbers. However, the temporal buffer size change might affect the communication rounds between the local models and the learning collaborator, which results in computation overhead.

5.4.5 Discussion

A noteworthy aspect of vertical federated learning is rebuilding the link between the distributed features. The typical approach uses identifiers shared by all the vertical partitions to reconstruct training samples for the global models (Hardy et al. (2017) and Nock et al. (2018)). In the case of

mobility count data, timestamps can be considered as the shared identifiers. However, even gathering the mobility counts in one place would not solve the forecasting problem since we still need to rebuild the temporal ordering between samples to predict future timestamps. Unlike MVFF, the comparison frameworks uniformly aggregate the learning from each vertical partition to build a forecasting model. However, as demonstrated in the experimental evaluation, this aggregation approach performs less than the MVFF framework. This improvement is due to the temporal and spatial synchronization modules that reconstruct the spatio-temporal correlation between timestamps and locations.

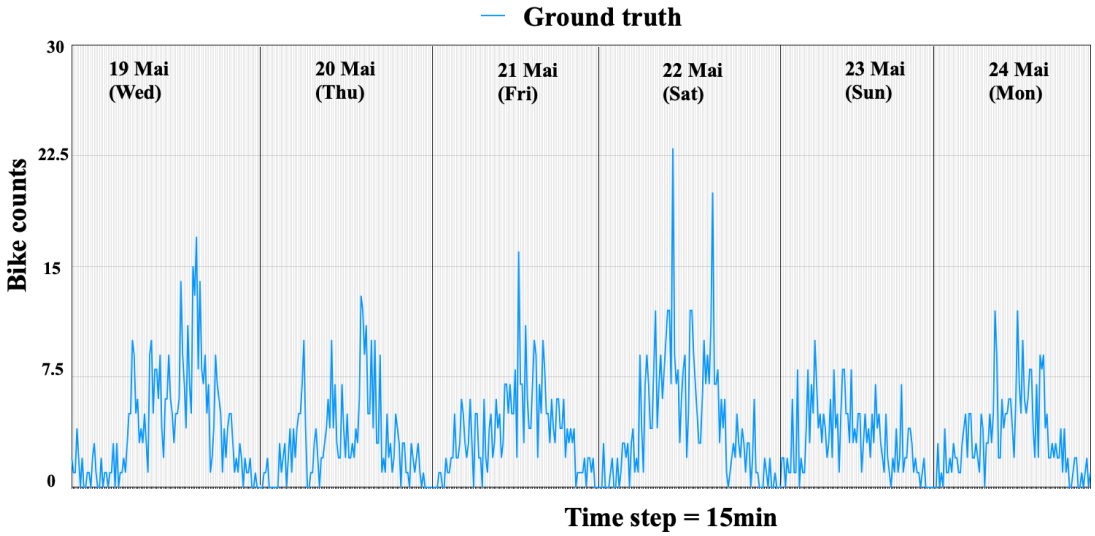
5.4.6 Threat to Validity

We evaluated our framework on two datasets with distinctive characteristics, as illustrated in Table 5.2. We also used state-of-the-art network models to implement the temporal and spatial synchronization between the models' parameters. By fixing the data's spatial and temporal resolution (the size of each region and the frequency at which the mobility aggregates are computed) for all experiments, we verified the framework's performance orthogonally to spatial and temporal resolution. However, if new recurrent network models were to be developed in the future, we would need to evaluate the framework with the new models' combination.

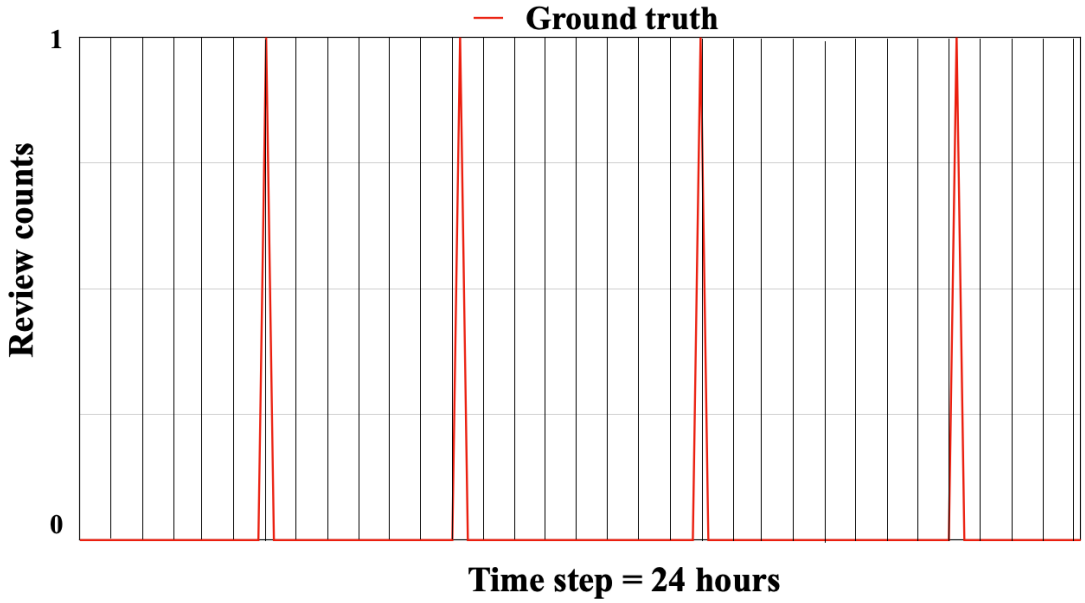
5.5 Summary

This chapter proposes a vertical federated learning framework for mobility data forecasting (MVFF), enabling the learning process to be jointly conducted over multiple organizations targetting a common location domain to produce accurate mobility predictions. Each organization extracts the embedded spatio-temporal correlation between its locations using a local learning model. A global model synchronizes the local models to incorporate the correlation between all the organizations' locations. We investigate the performance of MVFF under four variations of local and global models. We compare the MVFF's performance to two other federated frameworks on real-life datasets: New York Bike and Yelp reviews, achieving better performances. In most real situations, the organizations may cover irregular mobility time periods, leading to the disparity in data spread

(non-independent and non-identically distributed data). Moreover, we assume that communication with all organizations is reliable, leading to continuous model updates. However, differences in communication reliability capabilities are inevitable among organizations. Therefore, we plan to investigate the future integration of asynchronous federated learning strategies.

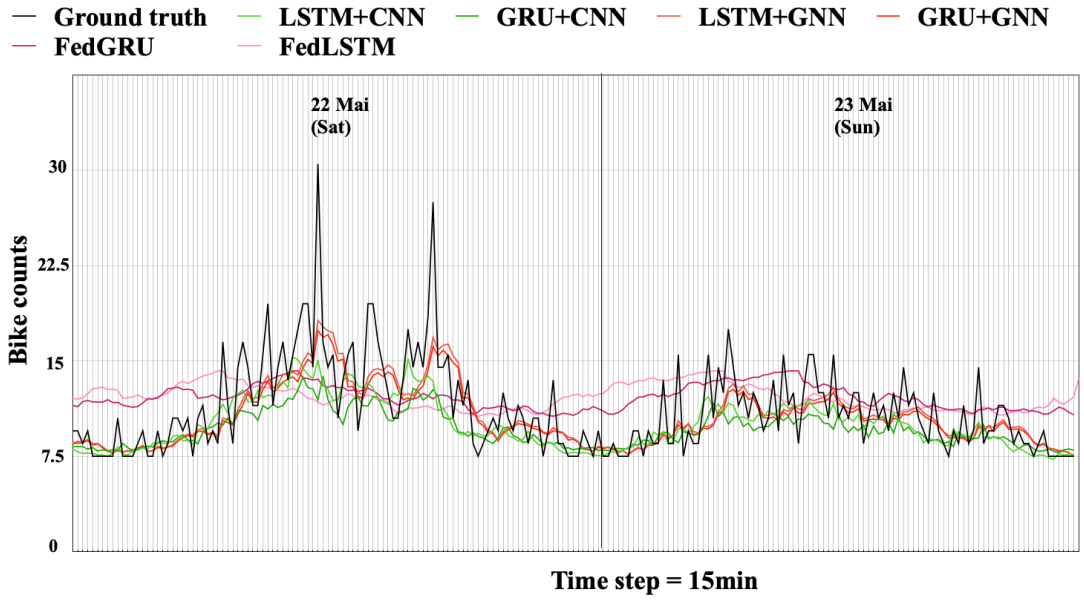


(a) True bike counts of 6 days

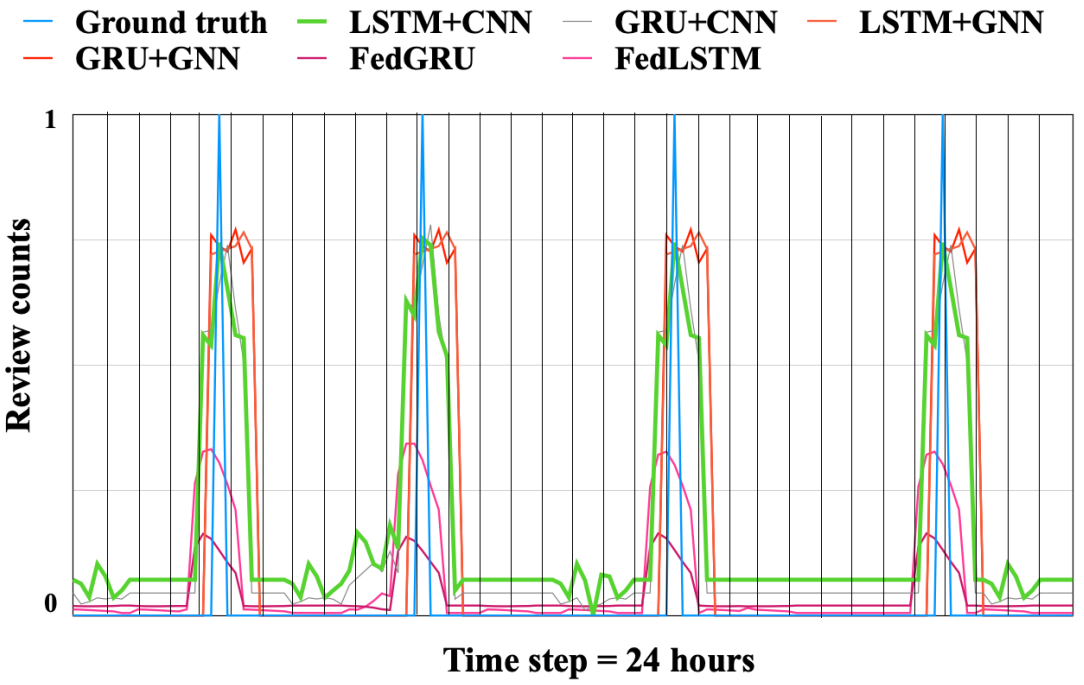


(b) True review counts of 40 days: Nov 1st, 2020 to October 11th, 2020

Figure 5.4: Mobility counts (ground truth) for Bike New York and Yelp review datasets.

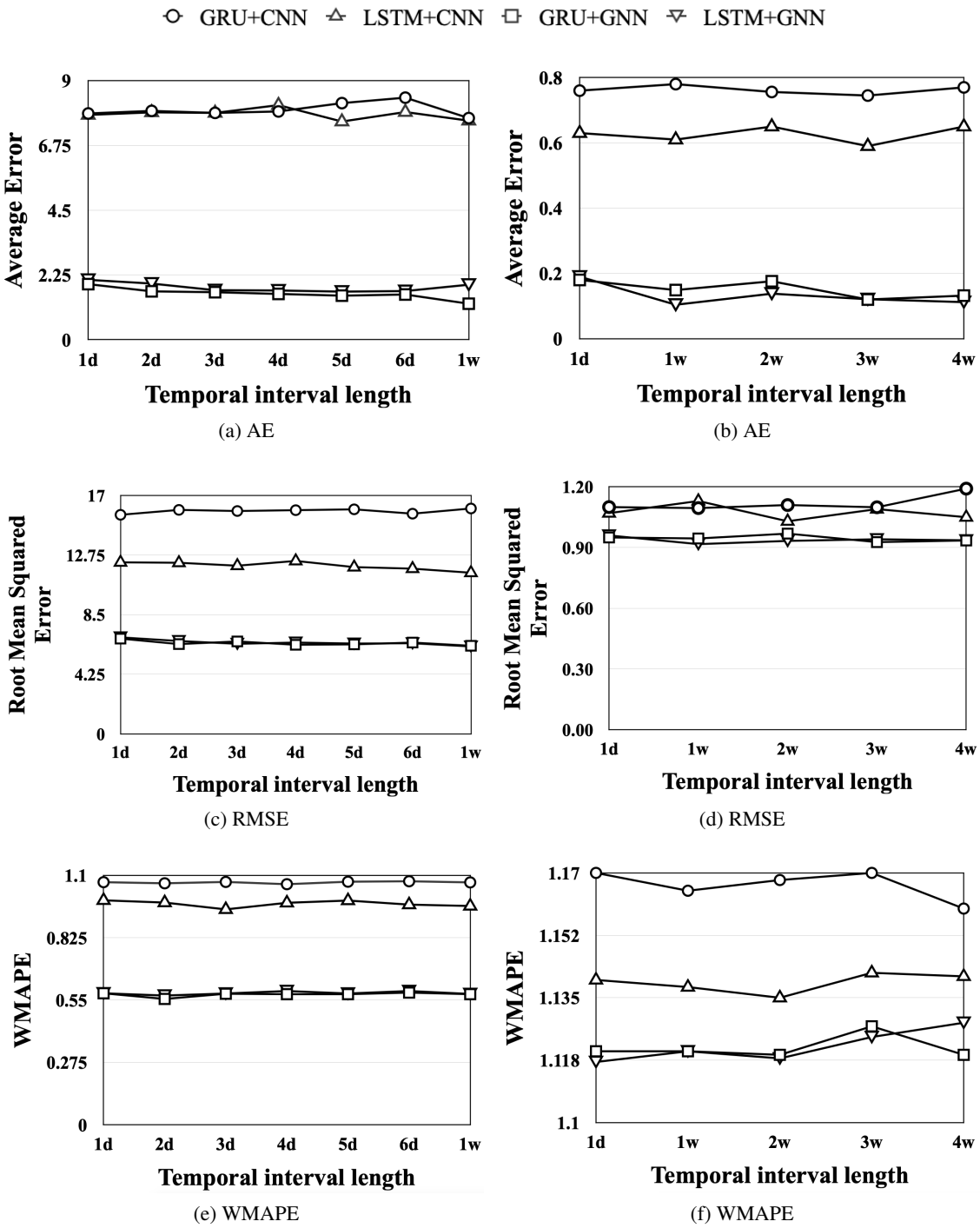


(a) Bike counts forecast of 2 days



(b) Yelp review forecast of 40 days

Figure 5.5: Mobility forecast for Bike New York and Yelp review datasets.



(a) AE

(b) AE

(c) RMSE

(d) RMSE

(e) WMAPE

(f) WMAPE

Figure 5.6: Metrics vs temporal interval length for the Bike New York and Yelp review datasets

Chapter 6

Adaptive Differential Privacy in Vertical Federated Learning for Mobility Forecasting

6.1 Introduction

There is a growing interest in exploiting data from multiple organizations to provide better knowledge acquisition. Vertical federated learning (Q. Yang et al. (2019)) is an emerging paradigm that leverages the collaboration between multiple organizations to build a learning model based on data in the form of isolated islands. However, this collaboration raises privacy concerns. For example, several works (Luo, Wu, Xiao, and Ooi (2021) and C. Fu et al. (2022)) show that fully trained models on sensitive features may leak information about the training data.

Numerous techniques aim at enhancing privacy during the learning process, including secure multi-party computation (R. Xu et al. (2021) and Y. Wu, Cai, Xiao, Chen, and Ooi (2020)), homomorphic encryption (Hardy et al. (2017) and Aono, Hayashi, Wang, Moriai, et al. (2017)), and differential privacy (Chaudhuri, Monteleoni, and Sarwate (2011) and Abadi et al. (2016)). Secure multi-party computation A. C.-C. Yao (1986) enables multiple organizations to collaboratively compute a common function, such as training a shared learning model, without revealing their raw data.

Homomorphic encryption (Gentry (2010)) allows computation on encrypted data while guaranteeing that the result is similar to the one based on raw data. Finally, differential privacy (Dwork et al. (2014)) rigorously quantifies the information disclosure of individuals participating in the dataset and masks it by introducing a level of uncertainty during the data processing.

Although secure multi-party computation and homomorphic encryption guarantee that the intermediate parameters exchanged between the organizations during the model's training are concealed (Aono et al. (2017) and Y. Wu et al. (2020)), they do not protect the privacy of the organizations' local training dataset users. Admittedly, several works (Carlini, Liu, Erlingsson, Kos, and Song (2019), Fredrikson, Jha, and Ristenpart (2015), and Hitaj, Ateniese, and Perez-Cruz (2017)) show that learning models may unintentionally memorize some training samples, raising a reasonable concern about the users' sensitive information leaks. By contrast, differential privacy can provide guarantees regarding the probability of reconstructing a sample in the training data by carefully adding random noise to the gradient at each training iteration (Stock, Shilov, Mironov, and Sablayrolles (2022) and Ghazi, Golowich, Kumar, Manurangsi, and Zhang (2021)).

Most of the proposed differential privacy solutions in federated learning (Truex, Liu, Chow, Gursoy, and Wei (2020) and Hu, Guo, and Gong (2021)) dedicate equal privacy budgets for the training data features. However, the privacy expectation of features from different organizations may vary widely G. Wang et al. (2019). For example, a bank may collaborate with a mobile network operator to build a better credit assessment model based on indicative features about their customers, such as phone bills and shopping logs (Han, Ding, Wang, and Huang (2019)). The privacy sensitivity of each of these features may be remarkably distinct. Therefore, we need to ask the following question:

RQ1: How to adjust privacy protection to the heterogeneous organizations' privacy needs?

Differential privacy solutions (Abadi et al. (2016) and Agarwal, Suresh, Yu, Kumar, and McMahan (2018)) add noise to the organization's gradients during training to protect each individual's privacy in the training dataset. Most solutions calibrate the added noise through a privacy budget that quantifies how much information is leaked by the differential privacy mechanism. A smaller budget indicates a lower bound privacy leakage corresponding to a more significant noise scale. In

addition, the privacy budget cumulates with each training iteration. Although the traditional differential privacy solutions protect the participant's privacy against several attack types (Stock et al. (2022) and Ghazi et al. (2021)), a uniform budget overlooks the dynamic changes in the model's convergence state during the training. For instance, when the model approaches the optimum, the gradient may be small in proportion to the amount of added noise. Therefore, a large noise might overwhelm the gradient's signal and jeopardize the model's convergence (Hong, Wang, and Zhou (2022)). Thus, an adaptive privacy budget allocation is preferable to a uniform allocation. Hence, the following research question:

> RQ2: How to allocate the privacy budget during training to prevent the participant's model privacy leak?

Motivated by these research questions, we present the Adaptive differential privacy for Vertical Federated Learning (AdaVFL) protocol that enforces the privacy protection of the dataset users by adjusting the privacy budget to their feature contribution while balancing privacy and utility via adaptive privacy budget allocation. We summarize our contributions below:

- We build a taxonomy based on several surveys to define the scope of the proposed protocol and evaluate the state-of-the-art privacy attacks and techniques (Section 6.2). We analyze the privacy risks based on the vertical federated setting, the threat model, and the scenario's assumptions.

- We design the Adaptive differential privacy for Vertical Federated Learning (AdaVFL) protocol involving the organizations and a centralized learning collaborator (Section 6.3.1). The protocol uses two components: a feature-aware privacy budget initialization and an adaptive budgeting scheme. The privacy budget initialization happens before the training process and aims at heterogeneously adjusting privacy protection to the organization's privacy-sensitive features (RQ1). The adaptive budgeting scheme allows each organization to allocate the privacy budget dynamically during training before the model parameters are exchanged between the organizations and the learning collaborator (RQ2).

- We propose a contribution-based method to evaluate the privacy susceptibility of each organization's features. We leverage the connection between the organization's training data features and the global model output, measured at the central learning collaborator. Then, we introduce two weighting strategies that heterogeneously adjust the privacy budget of each organization to the feature contribution metric (Section 6.3.2).

- We present an adaptive budget allocation strategy for the stochastic gradient descent that prevents the training dataset users' privacy leaks with a formal privacy guarantee. We adopt zero-Concentrated Differential privacy (zCDP) (Bun and Steinke (2016)), a generalization of differential privacy that better suits a larger number of iterations by inducing less noise for the same privacy guarantee. We leverage a feedback control mechanism to capture the model's convergence dynamics and adjust the consumed privacy budget accordingly (Section 6.3.3).

In addition, we implement AdaVFL on two real-world datasets (BikeNYC Gordon-Koven and Levenson (2014) and Yelp reviews Gordon-Koven and Levenson (2014)). We evaluate the resilience of the feature-aware budget initialization against the inference attack presented in Luo et al. (2021). Experimental results demonstrate that we decrease the attack's accuracy by up to 25% compared to the uniform feature budgeting. We also compare our algorithms to four state-of-privacy budget allocation strategies (Abadi et al. (2016), Lee and Kifer (2018), Yu et al. (2019), and Gong, Feng, and Xie (2020)). Experimental evaluation shows that our adaptive privacy budget increases accuracy by up to 15% on average compared to the state-of-the-art dynamic budget allocations (Section 6.4).

6.2 Privacy Threat Analysis

In this section, we first build a privacy threat taxonomy. We delimit the taxonomy based on the threat model, the learning paradigm, and our assumptions. The result determines the scope of our contributions and the related state-of-the-art review.

6.2.1 Privacy Threat Taxonomy

We propose a taxonomy of privacy threats in deep learning from two perspectives: the attacker and the target characteristics. We compile elements from several surveys (Q. Yang et al. (2019),

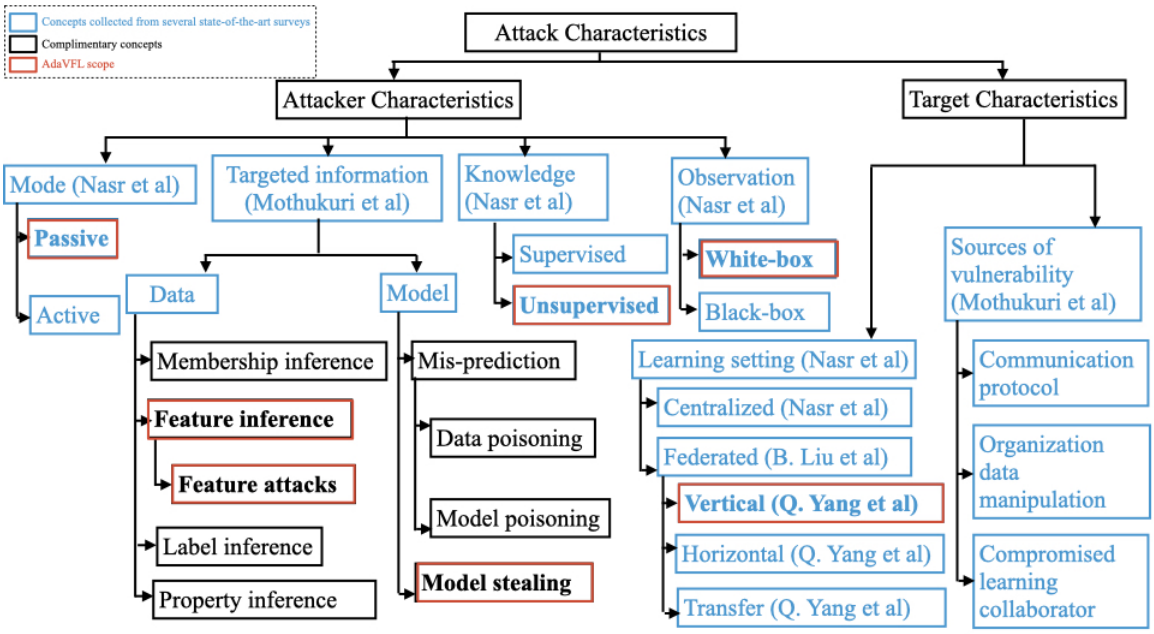


Figure 6.1: The privacy threat taxonomy details the characteristics of the attacker and target for privacy threat analysis.

Nasr, Shokri, and Houmansadr (2019), B. Liu et al. (2021), and Mothukuri et al. (2021)) and complement them with additional components, as illustrated in Figure 6.1.

The attacker's characteristics include the observation limit, which can be restricted to the model's output on arbitrary inputs (black box) or more intrusive, where the attacker can reproduce all the intermediate parameters of the model (white box). In the passive mode, the attacker can only observe the exchanged parameters computed by the training algorithm and the model. Conversely, the attacker can modify its parameters during training to influence the outcome in the active mode. The attacker's prior knowledge of the targeted training dataset is categorized into supervised, where it has a dataset subset or otherwise unsupervised.

The attacker's target information can be related to the training dataset or the learned model. For example, membership inference attacks aim at identifying users in the training dataset. Instead of targeting a single user, feature and label inference attacks try to reconstruct the training dataset attribute values. Finally, property inference attacks aim to learn details about the dataset, such as a distinctive feature distribution.

Several attacks, such as misprediction and parameter stealing, may target the organization's

model. An organization may feed the model false data or carefully manufactured parameters to skew the learning to its advantage. Moreover, the model parameters may be proprietary, such as stock price prediction; hence, the attacker's goal is to steal the model's parameters.

We categorize the target characteristics into learning settings and sources of vulnerability. The learning setting includes the centralized and the federated, which comprises vertical, horizontal, and transfer learning. Several sources of attack vulnerability are specified: the communication protocol between the organizations and the learning collaborator, the organization data manipulation, and the compromised learning collaborator. For example, a non-secure communication channel allows attackers to eavesdrop on the exchanged learning parameters. Moreover, a malicious organization might poison the model's training for its benefit. Finally, a compromised learning collaborator might be interested in reconstructing the private organization training datasets.

6.2.2 Threat Model

We assume that N organizations collaboratively train a deep learning model f in a vertical federated learning setting. Each organization, O_i, generates intermediate learning parameters using its local dataset, $\mathcal{D}_i$. This dataset comprises row entries with a feature set, S_i, and the combination of all dataset feature sets is P. The intermediate learning parameters of the local model, f_i, are sent to the learning collaborator, $\mathcal{C}$. When all parameters are received from every organization, $\mathcal{C}$ computes a training iteration of the global model f. This exchange continues until the global model reaches the desired convergence state.

In the threat model illustrated in Figure 6.2, the learning collaborator and the organizations are considered honest-but-curious adversaries (Paverd et al. (2014)). They aim to infer information about the target organization's training data and model parameters during training without deviating from the defined training protocol. This inference can be conducted by inspecting the exchanged learning parameters or the final learned model of the target organization.

6.2.3 Threat Analysis

This work targets collaborative learning in vertical federated learning, which excludes the remaining inapplicable characteristics in the learning setting. Since the learning collaborator collects

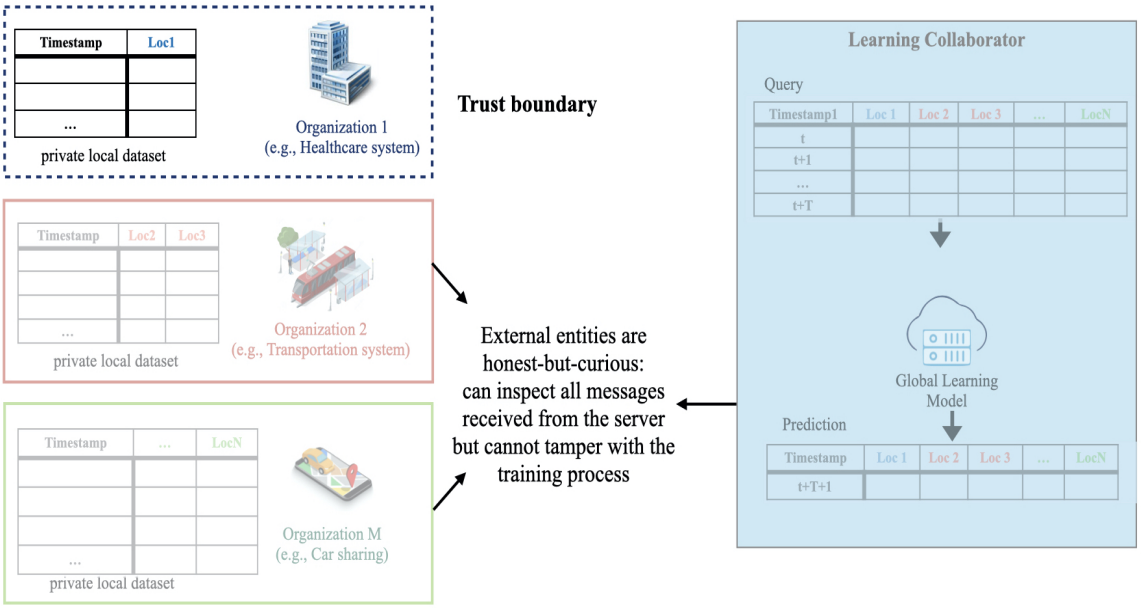


Figure 6.2: The adopted threat model: from the standpoint of each organization, other organizations and the learning collaborators are outside the trust boundary and considered honest-but-curious adversaries.

the organizations' model gradients and intermediate computation, we conclude that the attacker has a white-box access, as defined by Nasr et al. (2019).

Moreover, since we consider the organizations and the learning collaborator as honest-but-curious, the misprediction attacks targetting the learning model are out of scope. Additionally, this threat model eliminates the organization's data manipulation and the compromised learning collaborator as sources of vulnerability and active mode attacks.

We assume that the organizations have no prior knowledge about other organizations' training data. Therefore, following the definition presented by Nasr et al. (2019), the attacker's knowledge is unsupervised. Additionally, to allow the learning collaborator to train a global model that encapsulates all the organizations' features, it must know the features belonging to each organization. Therefore, our solution targets feature inference attacks where the attacker aims to reconstruct the participating organization's features.

Moreover, since several works illustrate the efficiency of differential privacy in reducing the membership leak risk (Bernau, Robl, Grassal, Schneider, and Kerschbaum (2021) and Farokhi and Kaafar (2020)), label inference attacks (Tang et al. (2022)), and dataset property inference attacks

(M. Chen and Ohrimenko (2023)), we consider these attacks outside the scope of this work. However, because of these attacks' privacy risk, we include experiments to test our proposed solution's performance against membership inference attacks in Section 6.6. Due to the lack of space, these experiments are limited to the membership inference attack (Nasr et al. (2019)).

Finally, we assume the communication channels between the organizations and the learning collaborator are secure. Consequently, we eliminate the communication protocol as a source of privacy vulnerability.

6.3 Adaptive Differential Privacy for Vertical Federated Learning Protocol Overview

In this section, we present the design overview of AdaVFL in Section 6.3.1. Then, we explain its components: (i) feature-aware privacy budget initialization to hinder the feature inference attacks highlighted in the taxonomy (Section 6.3.2) and (ii) adaptive budgeting scheme to protect the organizations' models against models stealing while improving the global model's accuracy (Section 6.3.3).

6.3.1 Overall Privacy Framework

The AdaVFL protocol adapts the privacy budget to the organization's feature contribution and the model's convergence. As illustrated in Figure 6.3, each organization, O_i, has a local model f_i that uses the private local dataset for training. The N organizations support the learning collaborator, $\mathcal{C}$, in training the global model, f, encompassing the total features, P. We assume that the local and global models do not share the same neural network architecture. This difference is a consequence of the vertical partitioning of the mobility dataset.

Moreover, the organizations' datasets hold different features used solely in their local models' training. Therefore, each organization is not aware of the features held by other organizations. Consequently, we factor in the feature contribution of each organization to the global model to compute the protection level required by the participant. The detailed approach is presented in Section 6.3.2.

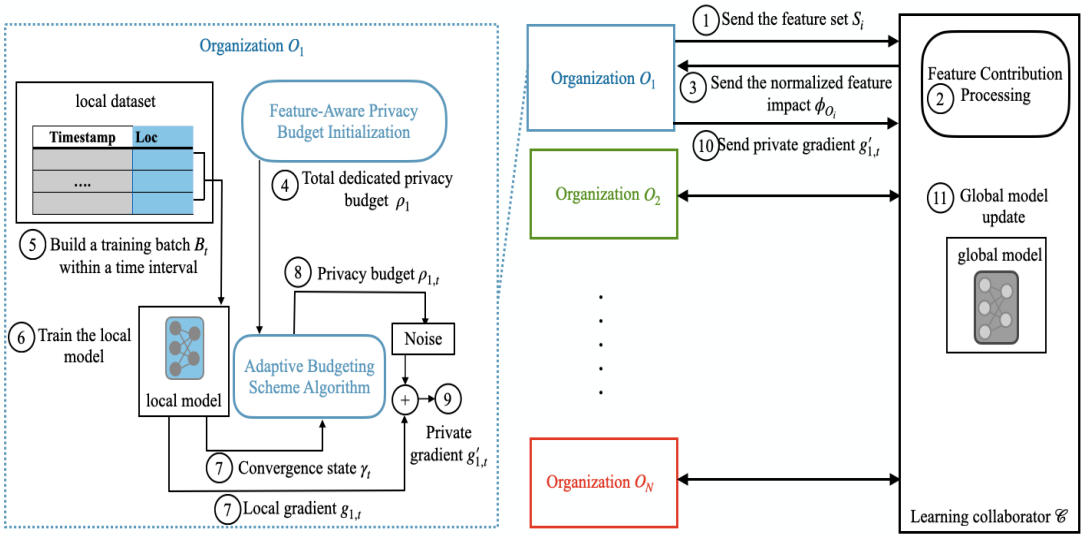


Figure 6.3: Illustration of the AdaVFL protocol components and steps. Each organization exchanges information about its features with the learning collaborator through the feature-aware privacy initialization and feature contribution processing components before the start of the training process. The organizations then assign a total privacy budget for the entire training task. Then, it breaks it down into individual privacy budgets per training iteration using the adaptive budgeting scheme component.

Additionally, we adapt the privacy budget assigned at each iteration in the training of the local model to prevent the privacy leakage of the organization's local dataset while improving the model's accuracy. Our approach is detailed in Section 6.3.3.

The high-level steps of the proposed protocol are shown in Algorithm 6. The collaborator and organizations initialize all their models' parameters before the learning starts. At the initialization, each organization, O_i, must send its feature set, S_i, to the collaborator. The collaborator then computes the organizations' feature contributions using a pre-trained model f_P and a publically available dataset $\mathcal{D}$ (line 8). Next, when the participant receives its feature contribution weight from the collaborator, it uses it to adjust the privacy protection level adequate to the learning task (lines 2 and 3). It is important to notice that sharing the feature set with the learning collaborator is a common approach in the state-of-the-art privacy-preserving techniques for vertical federated learning (D. Xu, Yuan, and Wu (2021) and Y. Wu et al. (2020)). Therefore, we do not adopt a privacy measure for sharing S_i.

The organization's model training, O_i, performs a training iteration of the local model with a

sampled batch and computes the private gradient using a privacy budget proportional to the variation of the accuracy of the local model (lines 4-7). The learning collaborator uses the private gradients to update the global model upon receiving the private gradients (lines 12-17). This collaboration continues until the organizations send a Halt signal to the learning collaborator because they consume their entire privacy budget or their models have converged.

Algorithm 6 Adaptive Differential Privacy in Vertical Federated Learning Protocol

Require: Feature set P, organizations' feature sets $\{S_i, i \in \{1, N\}\}$, pre-trained model f_P, public dataset $\mathcal{D}$

Ensure: The organizations' models $\{f_i, i \in \{1, N\}\}$, the global model f

1: **Initialize:** each organization model f_i parameters and global model parameters f
//Learning process at i-th organization
2: Receive the feature contribution weight ϕ_{O_i}
3: $\epsilon_i \leftarrow$ Feature-aware Privacy Budget Initialization (**Algorithm 8**)
4: **while** Training in progress **do**
5: $g'_{i,t} \leftarrow$ Compute private gradient (**Algorithm 9**)
6: Send $g'_{i,t}$ to learning collaborator $\mathcal{C}$
7: **end while**
//Learning process at collaborator $\mathcal{C}$
8: Feature Contribution Processing (**Algorithm 7**)
9: **for** Organization O_i in $\{O_i, i \in \{1, N\}\}$ **do**
10: Send the organization's feature contribution weight ϕ_{O_i}
11: **end for**
12: **while** Training in progress **do**
13: **while** Receiving ${g_{i,t}}'$ from organization O_i **do**
14: Compute private gradient g_c of global model f
15: Update global model f's parameter θ_c
16: **end while**
17: **end while**

6.3.2 Feature-Aware Privacy Budget Initialization

We individually evaluate the impact of each feature in the organization's feature set for an accurate estimation of the entire feature set's impact. This approach is inspired by the state-of-the-art

explainable artificial intelligence methods (Lundberg and Lee (2017)) in quantifying each participant's contribution in a learning model training.

The privacy vulnerability of each feature in the organization's training dataset depends on the nature of the global learning task. For example, if a location is deemed sensitive, it will require a high privacy guarantee through a high level of randomization during the training. However, the location may not affect the learning outcome of the global model, resulting in an unnecessary reduction of the global model's accuracy. Therefore, we should first assess each organization's feature privacy sensitivity. Then, we must design weighting strategies to initialize the privacy budget for the training based on the privacy assessment.

Feature Privacy Assessment

We characterize the privacy vulnerability of each feature with an impact metric based on its effect on the learning task. When the metric's value is significant, it becomes easier for a potential attacker to differentiate the presence of the feature in the training dataset. Consequently, tighter privacy protection is desirable. Moreover, based on the privacy threat analysis presented in Section 6.2, the learning collaborator and the organizations have no prior knowledge of any organization's targeted training dataset. Consequently, we conclude that the impact metric must be calculated without accessing the organizations' private datasets. We define the feature impact metric for a single feature as follows:

Definition 11 (Feature impact)**.** *We define the feature impact as the average value between the output of the model f w.r.t a feature j. Formally:*

$$\delta_j = Average_{X \in D} |f_P(X) - f_P(X_j^+)| \tag{21}$$

where f_P is a pre-trained global model that takes the feature set $P = \{S_i, i \in \{1, N\}\}$ as input. D is a publicly available dataset, and $X = \{x_1, ..., x_j, ..., x_p\}$ is a sample in D. $X_j^+ = \{x_1, ..., {x_j}^+, ..., x_p\}$ is a modified copy of X where ${x_j}^+$ is replaced by the randomly sampled value from the domain space of feature j.

Using a pre-trained model is a common feature assessment practice adopted in several centralized state-of-the-art learning works (N. Phan et al. (2017) and N. Phan et al. (2019)). Moreover, depending on the application domain, this dataset D can be rudimentary and easy to obtain in practice, such as the publicly available datasets (Geolife Zheng et al. (2010), Gowalla E. Cho et al. (2011), or T-Drive Yuan et al. (2010)). We assess the organization's feature set impact as follows:

Definition 12 (Normalized feature impact)**.** *The normalized feature impact is a value that summarizes the contribution of the features S_i of organization O_i in the global model learning process f, given the dataset D. Formally, the normalized feature impact value is:*

$$\phi_{O_i} = \frac{\sum_{k \in S_i} \delta_k}{\sum_{j \in P} \delta_j} \tag{22}$$

Algorithm 7 illustrates the computation of the contribution metric by the learning collaborator. It is important to note that the learning collaborator is responsible for verifying overlapping features between the organizations and sanitizing the feature set P before processing the feature impact metric. For each organization's feature, we iterate over the dataset samples and replace the sample's feature with a randomly selected value (line 4). This randomization can follow a shuffling or permutation scheme or a given distribution. Next, the resulting sample is processed through the pre-trained model f_P (line 5). The marginal value of the model's output and the expected original value output is calculated in line 6. Then, the feature impact is computed as the average difference value (line 8). Finally, the normalized feature impact is calculated based on each feature's sensitivity (line 10).

Weighting Strategies

After defining the normalized feature impact, we must design its correlation with the privacy budget computation. Intuitively, the lower the feature relevance, the looser the privacy should be. Conversely, the higher the feature relevance, the tighter the privacy should be. Moreover, as introduced in Definition 12, the contribution metric ranges from 0% to 100%. At 100%, the organization's data is the sole contributor to the global model. If the contribution metric's value is 0%, a potential attacker would be unable to differentiate between outputs where the organization's feature

Algorithm 7 Feature Contribution Processing

Require: Pre-trained model f_P, public dataset $\mathcal{D}$, overall feature set P, organizations' feature set $\{S_i, i \in \{1, N\}\}$

Ensure: The feature contributions of all organizations $\{\phi_{O_i}, i \in \{1, N\}\}$

for each participant O_i **do**
  for each feature j in P **do**
    for each sample $X = \{x_1, ..., x_j, ...x_n\}$ in $\mathcal{D}$ **do**
      $X_j^+ = \{x_1, ..., \hat{x_j}, ..., x_n\}$ Random sample features x_j
      $y^+ = f_P(X_j^+)$ Calculate the models result
      Calculate the marginal of the feature contribution at sample X: $F_X = f_P(X) - y^+$
    end for
    Compute the overall feature contribution value: $\delta_j = Average_{X \in D}|F_X|$ // *Definition* 11
  end for
  Compute the feature contribution weight for the participant $\phi_{O_i} = \frac{\sum_{k \in S_i} \delta_k}{\sum_{j \in P} \delta_j}$ // *Definition* 12
end for

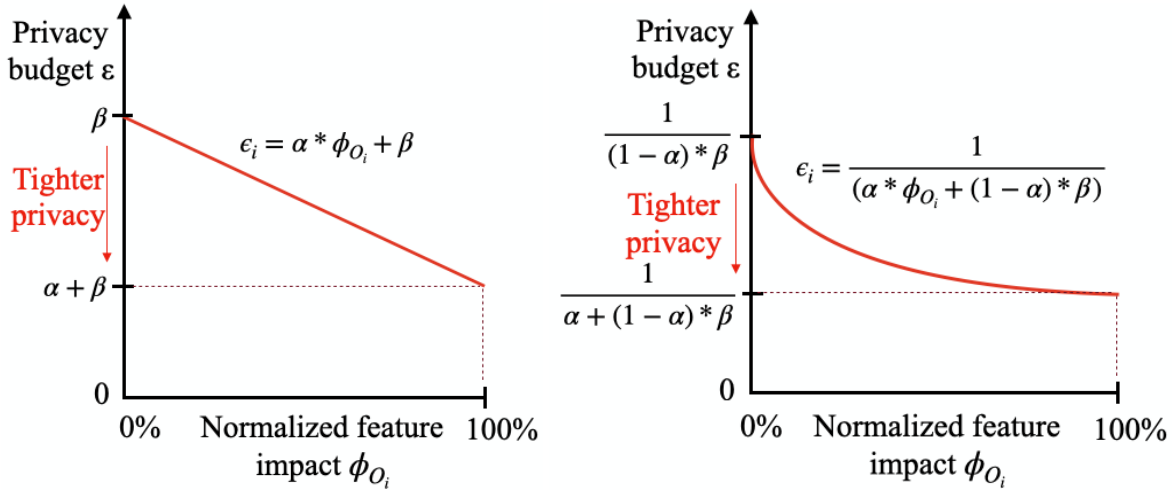
contributed or did not. Therefore, the privacy budget can be as large as possible. Considering all these aspects, we design two weighting strategies:

1) *Linear weighting*: We design a linear weighting function to allocate the privacy budget of each contribution based on feature impact. Inspired by the feature inference attack presented by Luo et al. (2021), the attacker's success rate accumulates as the organization's feature ratio increases. The linear weighting strategy is defined below:

$$W_{\alpha,\beta} = \alpha * \phi_{O_i} + \beta \tag{23}$$

where α controls the increasing rate of the privacy budget ϵ, and $beta$ represents the highest privacy budget assigned when the normalized feature impact is 0%.

2) *Reciprocal weighting*: The organization may use this strategy to strictly enforce the privacy guarantee for its feature by making the privacy budget inversely proportional to the normalized feature impact. This correlation is beneficial when a privacy-sensitive area needs higher privacy protection even with negligible participation in the global model. The reciprocal weighting strategy is described below:



(a) Linear weighting

(b) Reciprocal weighting

Figure 6.4: Weighting strategies that assign an initial privacy budget to each organization based on the normalized feature impact values, which range from 0% to 100%. The plots illustrate the weighting strategies with their limit values.

$$W_{\alpha,\beta} = \frac{1}{(\alpha * \phi_{O_i} + (1-\alpha) * \beta)} \tag{24}$$

where $0 < \alpha < 1$ and $0 < \beta < 1$. Additionally:

$$W_{\alpha,\beta} = \begin{cases} \epsilon = \frac{1}{(1-\alpha)*\beta} & \text{if } \phi_{O_i} = 0\% \\ \epsilon = \frac{1}{\alpha+(1-\alpha)*\beta} & \text{if } \phi_{O_i} = 100\% \end{cases} \tag{25}$$

Therefore, α controls the decreasing rate of the weighting strategy and β controls the minimum privacy budget value. Finally, Figure 6.4 illustrates the difference between the two strategies and Algorithm 8 illustrates the privacy budget computed by each organization.

Algorithm 8 Feature-aware Privacy Budget Initialization

Require: Organization feature weight ϕ_{O_i}, the weighting strategy parameters α and β

Ensure: Total privacy budget,ϵ, for organization O_i

1: Send the organization's feature set S_i

2: When the feature participation weight ϕ_{O_i} is received:

3: Compute the adaptive privacy budget $\epsilon = W_{\alpha,\beta}(\phi_{O_i})$

6.3.3 Adaptive Budgeting Scheme Algorithm

Once each organization, O_i, determines the overall privacy budget ϵ_i to dedicate to the training task, it needs to allocate a portion to each iteration. We propose an adaptive budgeting scheme that uses a larger share of the privacy budget (less noise) when the model's accuracy increases and a smaller share (more noise) otherwise. Therefore, this dynamic budget allocation improves utility by reducing the noise scale added to the model's gradient. To achieve this dynamic budget allocation, we need to answer the following research question:

> RQ2.1: How to allocate the privacy budget at each training iteration in correlation with the model's convergence?

The first challenge in designing the adaptive budgeting scheme is determining the convergence dynamic of the learning model. Our solution uses the training loss, γ_t, computed at each iteration, t, to indicate the model's convergence state. We call this metric the convergence index. Since the convergence index may not improve monotonically as the training progresses, we use the moving average to reduce unnecessary fluctuations. Therefore, we define the average convergence index, γ_a, as the averaged values throughout m training iterations as follows:

$$\gamma_a = \frac{1}{m} \sum_{k=a-m+1}^{m} \gamma_k \tag{26}$$

The second challenge is in correlating the convergence dynamics to the privacy budget. We leverage a feedback control mechanism (Bellman (1964)) to adjust the privacy budget to the model's convergence dynamics. Feedback control loops use measurement information to adapt a variable to achieve the desired result. As illustrated in Figure 6.5, the convergence index at each iteration represents the loop's measurement information (step 1). This input is then compared with a reference, γ_{min}, in step 2. This reference can be the same as the threshold value at which the model training stops because the model has converged. We use γ_{min} as a denominator in Equation 27 to restrain the feedback error magnitude. Consequently, we avoid overconsuming the privacy budget, leading to prematurely stopping the model's training. The comparison result constitutes the feedback error

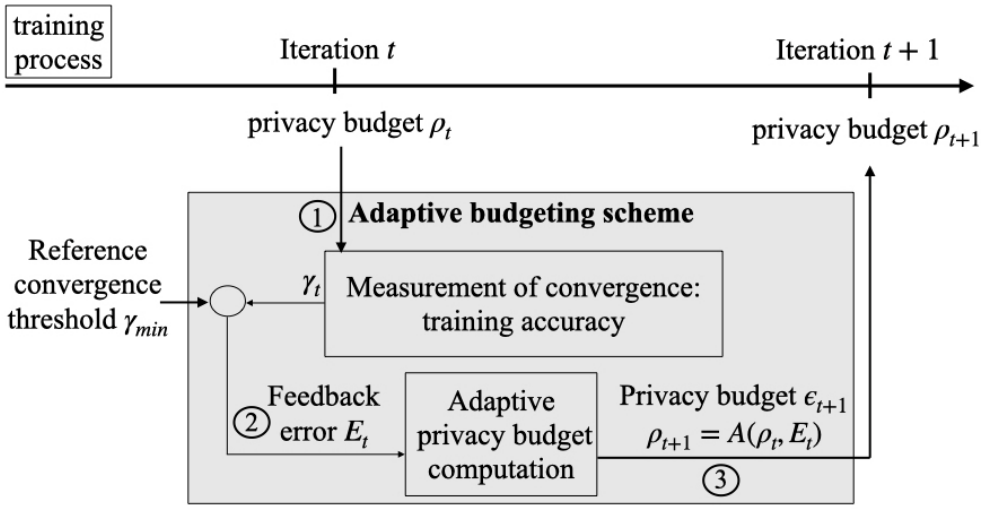


Figure 6.5: The feedback control loop adjusts the privacy budget ρ_t at each training iteration t to the model's convergence dynamics through the feedback error E_t. The convergence of the model is assessed with the training loss.

E_t. Finally, the desired result of the feedback control loop is adjusting the privacy budget to the model's convergence dynamics through the function A in step 3. Therefore, we define the feedback error as follows:

$$E_t = \frac{\gamma_t - \gamma_a}{\max\{\gamma_t, \gamma_{min}\}} \tag{27}$$

Finally, the last challenge in adjusting the privacy budget to the convergence error is formulating the correlation between the two. Therefore, we need to answer the following question:

> RQ2.2: How to regulate the effect of the convergence state on the dynamic privacy budget allocation?

Our goal is to reduce the added noise distribution the model converges. Therefore, we design the correlation function, A, that increases the privacy budget with the convergence index, resulting in a smaller noise scale. Moreover, the correlation function has a weight, μ, that sets the effect of the feedback error on the newly calculated budget. The correlation function is defined below:

$$A(\rho_t, E_t) = \rho_t * (1 + \mu * E_t) \tag{28}$$

Algorithm 9 Adaptive Budgeting Scheme Algorithm

Require: Total privacy budget (ϵ_i,δ_i), clipping threshold C_i, learning rate η, local dataset $\mathcal{D}_i$, local model f_i, reference training accuracy γ_{min}, update interval m

Ensure: Private gradient g_t per iteration t, the local model f_i's parameters w_i

Initialize the total privacy budget ρ_i //(*Equation* 10)
Initialize the consumed budget $\rho_c = 0$
for $t = 0....T$ **do**
    Sample a batch B_t from the local dataset $\mathcal{D}_i$
    Compute and clip gradient:
    for $X_j \in B_t$ **do**
        Compute $g_t(X_j) = \nabla\mathcal{L}(w_i^t, X_j)$
        Clip $g_t(X_j) \leftarrow g_t(X_j)/max(1, \frac{\|g_t(X_j)\|_2}{C_i})$
    end for
    Compute convergence index γ_t
    if $t = 0$ **then**
        $\rho_t = \rho_i/T$
    else
        if $\rho_c < \rho_i$ **then**
            if t is a multiple of m **then**
                Compute the averaged convergence index γ_a // (*Equation* 26)
                Compute the privacy budget: $\sigma_t = \sigma_{t-1}$
            else
                Compute the convergence dynamic E_t // (*Equation* 27)
                Compute the privacy budget: $\sigma_t = \frac{1}{2\sqrt{A(\rho_{t-1}, E_t)}}$ // (*Equation* 28)
            end if
            Add noise: $g_t' = \frac{1}{|B_t|}(\sum_{X_j \in B_t} g_t(X_j) + \mathcal{N}(0, \sigma_t^2 C_i^2 \mathbb{I}))$
            Send g_t' to the learning collaborator
            Calculate the consumed privacy budget $\rho_c \leftarrow \rho_c + \rho_t$
            Descent $w_i^{t+1} \leftarrow w_i^t - \eta g_t'$
        else
            Send Halt to the learning collaborator
            Break;
        end if
    end if
end for

Algorithm 9 describes the training process at each organization O_i. Since we are using z-CDP, the first step is to convert the privacy budget (ϵ_i,δ_i) to ρ_i (line 1) following the Equation 10 introduced in Chapter 2. The value of δ_i is less than the inverse of the organization's dataset size polynomial, based on the guidelines by Dwork et al. (2014). Next, we initialize the cumulative privacy budget to keep track of the budget consumption during the training (line 2). At each iteration, the algorithm samples a batch B_t from the local dataset, computes the gradient, and clips it with the clipping threshold C_i (lines 4-9). The convergence index γ_t, which can be the training loss, is also calculated on the batch B_t (line 10). If the privacy budget is not exhausted, the noise scale σ_t is calculated based on ρ_t following Equation 11 (line 12). The initial value of ρ_t is obtained by uniformly dividing the whole budget ρ_i over the total number of iterations T (line 12). In the later iterations, the budget is computed following the feedback loop (lines 15-21). The training terminates when the privacy budget is depleted or the training goal is achieved.

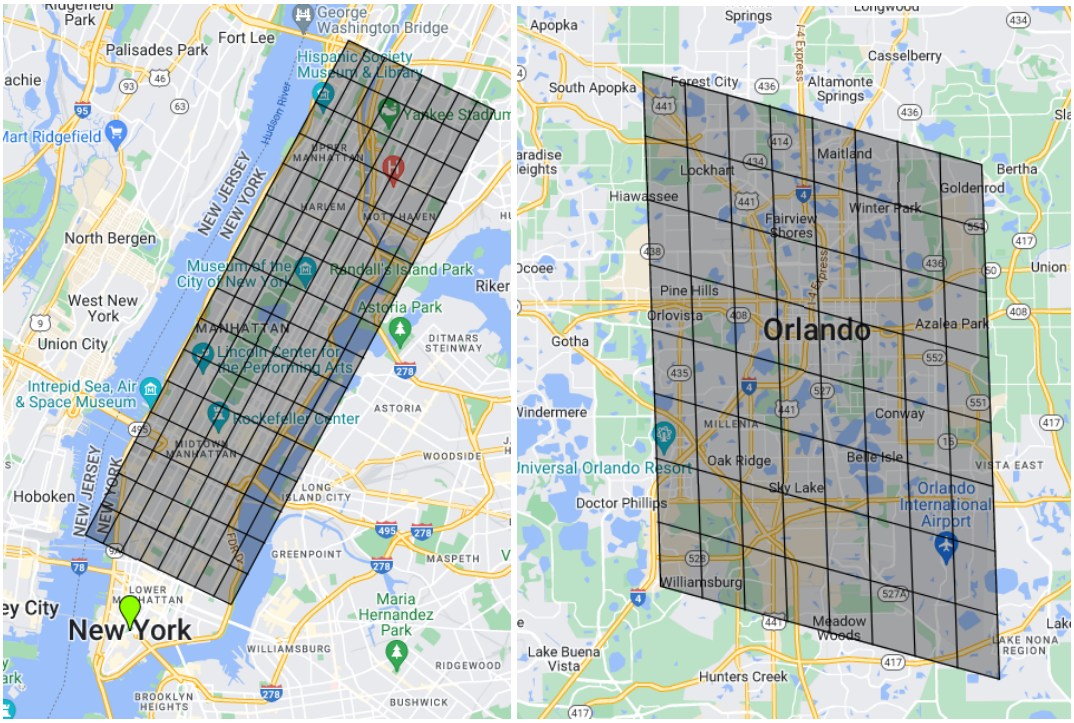
Algorithm 9 targets the model stealing attacks highlighted in Figure 6.1. To illustrate its privacy guarantee, we introduce the following theorem:

Theorem 8. *Algorithm 9 satisfies (ϵ_i,δ_i)-differential privacy.*

Proof. Based on the definition of Zero-Concentrated Differential Privacy (Definition 7, Algorithm 9 satisfies ρ_t-zCDP at each iteration t. Since ρ-zCDP also satisfies the composition theorem (Theorem 1), then Algorithm 9 satisfies $\sum_t \rho_t$-zCDP. The training process stops when the privacy budget ρ_i is exhausted. Therefore, $\sum_t \rho_t = \rho_i$. Moreover, based on Equation 10, we conclude that algorithm 9 satisfies (ϵ_i,δ_i)-differential privacy. □

6.4 Case Study: Mobility Forecasting for Bike New York and Yelp Reviews

To evaluate AdaVFL, we integrate its components into the framework for mobility forecasting in vertical federated learning in Chapter 5. This section describes the case study, datasets, metrics, and the state-of-the-art to compare our algorithms. Finally, we present the design and results of the experiments.



(a) New York City Bike dataset (b) Yelp review dataset

Figure 6.6: Grid division into 16x8 regions for the New York bike dataset and 8x8 regions for the Yelp review datasets. Each region has timestamped mobility counts and corresponds to a single feature.

6.4.1 Features and Data Processing

The mobility forecasting framework presented in Chapter 5 follows the ensemble learning paradigm, where each organization has its learning model and dataset, and the organizations collaborate to train a global model. It relies on a learning collaborator to orchestrate the training between the organizations. It provides two choices of neural network architectures implementations for the organizations: LSTM and GRU, and two for the global model: GNN and CNN. We select the combination of network architectures that renders the highest utility for the privacy evaluation of our proposed algorithm. Therefore, the organizations' local models are implemented as GRU, and the global model as GNN. We implement AdaVFL and the comparison frameworks using Pytorch Paszke et al. (2019) and the Colab platform.

We use two large-scale real-world datasets from New York City Gordon-Koven and Levenson (2014) and Yelp reviews Yelp (2019). The features of each organization are defined using regions

introduced below:

Definition 13 (Region (S. Wang, Cao, Chen, Peng, and Huang (2020))). *Given a city partition of an mxn grid map, based on the longitude and latitude, the $i-th$ row and $j-th$ column represent a region. A grid is defined as the region set $R = \{r_{1,1}, ..., r_{i,j}, ..., r_{m,n}\}$.*

Each feature is linked to a region, as illustrated in Figure 6.6; it consists of the mobility count captured over consecutive timestamps as follows:

New York City Bike dataset Gordon-Koven and Levenson (2014): It contains monthly trip details of bikes based on the bike stations deployed in New York City. We split the city into 16x8 regions, each containing bike stations based on their latitude and longitude, and each region represents a feature.

Yelp review dataset Yelp (2019): It contains 192k businesses from metropolitan areas in the United States with their locations, and 6.68M timestamped users' reviews. We divide the map into a grid of 8x8 regions, each containing businesses based on their location, and each region represents a feature.

We implement a MapReduce (Dean and Ghemawat (2008)) framework[1] to transform the two datasets into time series. The mapping task reads the timestamped bike ride or business review with its corresponding bike station or business location and determines its corresponding region based on the latitude and longitude. Upon receiving the region identifiers, the reduce task sorts the associated timestamps and counts them based on a fixed interval, resulting in a time series. The New York City Bike uses 15 minutes, and Yelp uses 6 hours. We collect the first four months of 2021 for New York City Bike: February 1st to Mai 31st, 2021, resulting in a dataset of 1.5 million timestamps. The review dates for Yelp reviews start on January 1st, 2018 and stop on December 31st, 2020. Overall, the sequence dataset contains 280k timestamps.

6.4.2 Comparison Frameworks and Metrics

We evaluate AdaVFL from the privacy and utility perspectives. Therefore, we compare the privacy metric, MSE, with the results of the following attack:

[1] https://github.com/FatimaErrounda/datatransformation

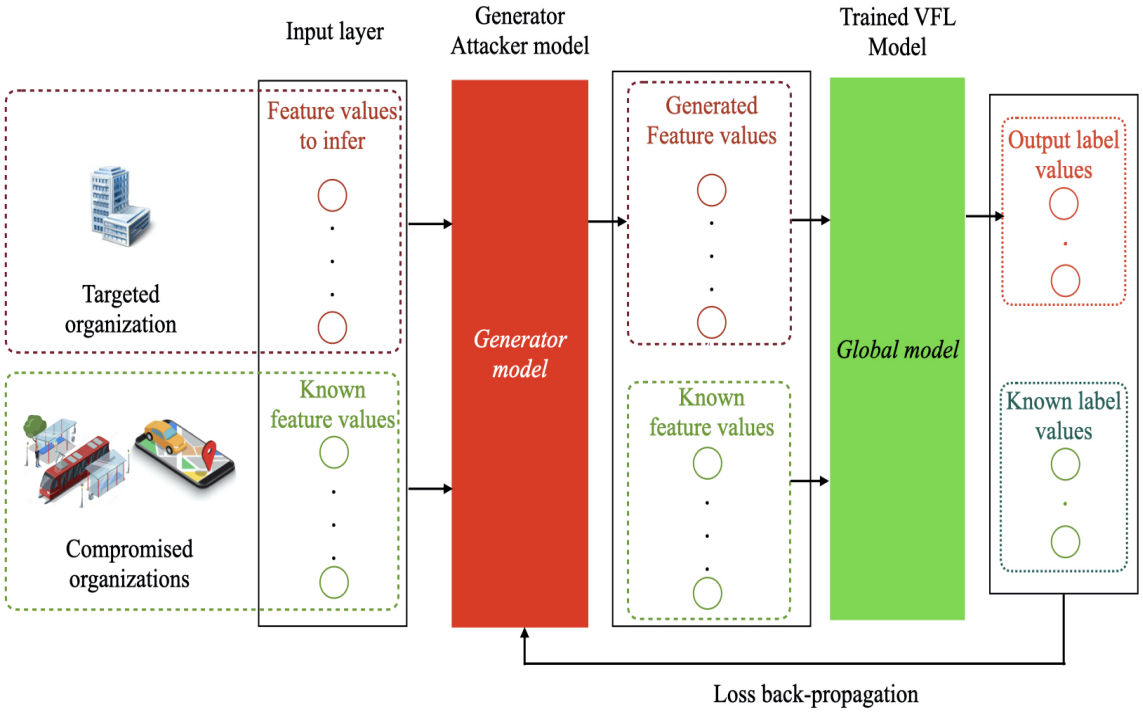


Figure 6.7: The privacy feature inference attack details: The attacker trains a model that takes the known feature values from compromised organizations and randomly generated features for the targeted organization as input. Then, the attacker model generates potential feature values. The loss is the comparison result between the targeted trained model's output given the guessed features and the true labels.

- **Feature inference attack** (Luo et al. (2021)): This attack targets features of a participant in vertical federated learning. As illustrated in Figure 6.7, the attacker builds a generator model that takes the attacker's known feature values and a set of random variables for the targeted participant features as input. The generator model is trained to estimate the targeted feature values that minimize the loss between the estimation and the ground-truth feature values.

For the utility evaluation, we compare the adaptive budgeting scheme with the following:

- **DP-AGD** (Lee and Kifer (2018)): This solution divides the privacy budget at each iteration into two portions. The first portion is used to compute the noisy gradient, while the second checks if the noisy gradient descends the model to a converging direction or the opposite. When the new descent direction is deemed not beneficial to the model's convergence, DP-AGD increases the privacy budget used to compute the gradient to reduce the added noise scale.

- **Adaptive budget based on validation accuracy** (Yu et al. (2019)): This solution increases the privacy budget by a fixed factor, k, whenever the validation accuracy during training is below a threshold. The validation accuracy is calculated using a small publicly available dataset from the same distribution as the validation dataset.

- **Interval increase strategy** (Gong, Feng, and Xie (2020)): This work proposes a strategy that consistently increases the budget to a maximal value with uniform steps proportionally to the current epoch or iteration. The new budget is then used to perturb the gradient.

- **Uniform DP-SGD** (Abadi et al. (2016)): This baseline solution assigns a uniform budget to each training iteration and uses it to compute the noisy gradient.

We use the mean square error (MSE) per feature for the privacy evaluation to measure the overall accuracy in reconstructing multiple targetted features. The mean square error is computed as below:

$$MSE = \frac{1}{n * d} \sum_{i=1}^{n} \sum_{j=1}^{d} (X_j^i - \hat{X}_j^i)^2 \tag{29}$$

where n is the number of samples in the prediction sample dataset, d is the number of targetted features, $\hat{X}_j^i$ and X_j^i are the inferred j-th feature values and the ground-truth of the i-th sample, respectively.

We choose three metrics for utility assessment: root mean square error (RMSE), weighted mean absolute percentage error (WMAPE), and average error (AE). The metrics' calculation is as follows: let X_l be the mobility count at a timestamp l and X_l' be the corresponding predicted count, where l varies from 1 to n, consequently:

$$RMSE = \sqrt{\frac{1}{n} \sum_{l=1}^{n} (X_l - X_l')^2} \tag{30}$$

$$WMAPE = \frac{\sum_{l=1}^{n} |X_l - X_l'|}{\sum_{l=1}^{n} |X_l|} \tag{31}$$

$$AE = \frac{1}{n} \sum_{t=1}^{n} (x_t - x_t') \tag{32}$$

6.4.3 Experiment Design and Results

We design five experiments: one to evaluate the first research question (RQ1) from the privacy and utility perspectives and four to assess the utility of AdaVFL (RQ2). We repeat all the experiments ten times and report the average. Our code is available at https://github.com/FatimaErrounda/AdaVFL. We introduce the experiments and results below:

Experiment 1 (RQ1) Assesses how the heterogeneous privacy protection through the feature-aware privacy budget initialization affects the resiliency against the privacy attack Luo et al. (2021).

Setup: Vary the ratio of features that the attacker attempts to reconstruct, from 10% to 90%, and capture the privacy metric, MSE. After computing each feature's impact metric, sort them in increasing order and select the highest as known by the attacker. The attacker's model must guess the remaining features. Run the privacy attack on a model trained: (1) without privacy guarantee, (2) with a uniform privacy budget per organization, and (3) with our feature-aware budget initialization. We ran the experiment 10 times and reported the average.

Results: Table 6.1 and Table 6.2 respectively illustrate the results of the privacy attack success for Bike New York and Yelp review datasets. We observe that the reciprocal weighting strategy results in a smaller attack success rate (larger MSE) than the linear weighting strategy. This improvement in resilience is due to the tighter privacy guarantee provided by the reciprocal weighting strategy through a smaller privacy budget. Moreover, both strategies surpass the uniform budget initialization, hindering the attacker's feature inference attack. Additionally, the MSE values are higher than the reported values by Luo et al. (2021). This difference is because the targetted model by Luo et al. (2021) is a classifier with a simple network architecture compared to our Graph Neural Network model. Furthermore, the targetted features in our experiments are sequential mobility data, whereas, in Luo et al. (2021), they are not.

Experiment 2 (RQ2.1) Assesses the effect of the correlation between the adaptive budgeting scheme and the model's convergence on utility.

Setup: We fix the number and epochs to 16 and 10 for the Bike and Yelp datasets, respectively. We capture the forecasting performance using the utility metrics for the comparison solutions and

Feature ratio in %	Non-DP	Uniform-DP Abadi et al. (2016)		Linear weighting		Reciprocal weighting	
	MSE	MSE	+%	MSE	+%	MSE	+%
10	0.204 ± 0.01	0.214 ± 0.01	4.77	0.224 ± 0.02	**9.60**	0.229 ± 0.02	**12.33**
20	0.209 ± 0.01	0.222 ± 0.01	6.45	0.224 ± 0.02	**7.04**	0.230 ± 0.02	**9.93**
30	0.216 ± 0.02	0.231 ± 0.02	6.72	0.237 ± 0.03	**9.37**	0.0239 ± 0.03	**10.21**
40	0.228 ± 0.02	0.230 ± 0.01	1.18	0.239 ± 0.02	**4.99**	0.248 ± 0.2	**8.86**
50	0.237 ± 0.01	0.255 ± 0.04	7.60	0.258 ± 0.04	**8.67**	0.265 ± 0.02	**11.51**
60	0.235 ± 0.07	0.246 ± 0.03	4.90	0.252 ± 0.02	**7.26**	0.257 ± 0.02	**9.26**
70	0.446 ± 0.03	0.469 ± 0.02	5.17	0.478 ± 0.04	**7.14**	0.491 ± 0.03	**10.14**
80	0.708 ± 0.07	0.761 ± 0.03	7.50	0.77 ± 0.06	**8.68**	0.783 ± 0.03	**10.60**
90	1.030 ± 0.12	1.108 ± 0.04	7.57	1.115 ± 0.1	**8.31**	1.135 ± 0.09	**10.19**

Table 6.1: The attack's success rate (MSE) comparison for the Bike New York dataset, the higher the value, the harder it is for the attacker to reconstruct the targeted features. (+%) represents a percentage increase in the attack's average metrics compared to the no-privacy solution.

Feature ratio in %	Non-DP	Uniform-DP Abadi et al. (2016)		Linear weighting		Reciprocal weighting	
	MSE	MSE	+%	MSE	+%	MSE	+%
10	0.194 ± 0.7	0.219 ± 0.1	13.14	0.226 ± 0.2	**16.60**	0.238 ± 0.3	**22.67**
20	0.201 ± 0.1	0.203 ± 0.06	1.19	0.212 ± 0.08	**5.64**	0.214 ± 0.05	**6.61**
30	0.201 ± 0.1	0.204 ± 0.09	1.08	0.208 ± 0.1	**3.25**	0.221 ± 0.2	**9.79**
40	0.214 ± 0.2	0.237 ± 0.04	10.86	0.243 ± 0.02	**13.54**	0.248 ± 0.1	**15.85**
50	0.242 ± 0.01	0.251 ± 0.03	3.80	0.255 ± 0.04	**5.54**	0.276 ± 0.15	**14.36**
60	0.407 ± 0.05	0.482 ± 0.08	18.46	0.510 ± 0.2	**25.40**	0.511 ± 0.1	**25.68**
70	0.415 ± 0.1	0.490 ± 0.02	17.91	0.519 ± 0.2	**25.03**	0.521 ± 0.2	**25.59**
80	0.669 ± 0.08	0.776 ± 0.02	15.92	0.792 ± 0.08	**18.32**	0.831 ± 0.02	**24.17**
90	1.206 ± 0.05	1.253 ± 0.08	3.81	1.258 ± 0.05	**4.27**	1.283 ± 0.1	**6.37**

Table 6.2: The attack's success rate (MSE) comparison for the Yelp reviews dataset, the higher the value, the harder it is for the attacker to reconstruct the targeted features. (+%) represents a percentage increase in the attack's average metrics compared to the no-privacy solution.

the adaptive budgeting scheme algorithm.

Results: As illustrated in Figures 6.8 and 6.9, AdaVFL gives the lowest error values across the three utility metrics for the local and the global models. The uniform solution (Abadi et al. (2016)) performs the worst out of the five solutions for the global and the local models by giving the highest error values. The validation accuracy solution (Yu et al. (2019)) achieves lower error values than the uniform solution. However, relying on a threshold to decrease the added noise scale (increase the privacy budget) leads to maintaining the same privacy budget for several epochs, which results in a similar outcome as the uniform solution. Conversely, AdaVFL adjusts the noise scale following the model's convergence state without using a fixed threshold. The interval increase strategy (Gong, Feng, and Xie (2020)) performs better than the validation accuracy solution (Yu et al. (2019)) because it decreases the noise scale periodically regardless of the model's convergence, hence avoiding the pitfall of sustaining a uniform noise scale for several epochs unnecessarily. Nevertheless, due to its lack of adaptability to the model's convergence state, the continuous increase strategy performs less than AdaVFL. Moreover, DP-AGD (Lee and Kifer (2018)) divides the privacy budget into two portions at each iteration, which might lead to a more significant added noise to the gradient, hindering the learning performance.

Experiment 3 (RQ2.1) Assesses the effect of the correlation between the adaptive budgeting scheme and the model's convergence on privacy.

Setup: We fix the total privacy budget ($\epsilon = 1.6$) and capture the utility of the comparison frameworks and the adaptive privacy budget allocation.

Results: Figures 6.10 and 6.11 demonstrate that AdaVFL achieves the highest utility overall of the local and global models. To better understand the budget consumption of the local model, we captured the privacy budget per iteration in Figure 6.12. We notice that despite the uniform (Abadi et al. (2016)) solution training the largest number of epochs for the fixed budget, its performance is less than AdaVFL. This difference stems from maintaining an identical noise scale through all the epochs while AdaVFL adjusts the added noise range. On the other hand, DP-AGD (Lee and Kifer (2018)) is the first solution to exhaust the total privacy budget because of its division at each iteration to determine the benefit of the noisy gradient for the model's convergence. AdaVFL achieves the

same goal without consuming additional budget, dedicating more to the actual gradient perturbation. Moreover, unlike the interval increase strategy (Gong, Feng, and Xie (2020)), AdaVFL reduces the noise scale to a smaller value faster, leading to a higher utility. Finally, the validation accuracy solution (Yu et al. (2019)) maintains an identical noise scale for multiple iterations because the validation test does not reach the assigned threshold. In contrast, AdaVFL reduces the added noise scale at earlier iterations, reaching a better utility.

Experiments 2 and 3 show that we perform better than state-of-the-art solutions. However, we want to learn more about the convergence speed of our solution. There are two typical methods in the literature to ensure that a learning model will eventually converge: (1) theoretical analysis (K. Wei et al. (2020), Huang, Hu, Guo, Chan-Tin, and Gong (2019), and Pichapati, Suresh, Yu, Reddi, and Kumar (2019)) and (2) data-driven empirical experiments (Tramer and Boneh (2020), Frigerio, de Oliveira, Gomez, and Duverger (2019), and H. Liu et al. (2021)).

The closest theoretical analysis to our setting is the convergence upper bound for federated learning presented by K. Wei et al. (2020). Based on the assumption that the loss function of each client is convex, β-Lipschitz, L-Lipschitz smooth, and satisfies the Polyak-Lojasiewicz condition, it presents a convergence upper bound for differentially private federated learning (Theorem 2 in K. Wei et al. (2020)). The upper bound depends on the privacy budget, ϵ, the clipping bound, C, β, L, and other parameters. Therefore, for a given model to maintain the same convergence with different privacy budgets, other algorithm parameters must be adjusted.

Wei et al. (K. Wei et al. (2020)) use simulation to estimate the values of β and L, then evaluate the learning performance with varying levels of protection (ϵ) using a Multi-Layer Perception network. We follow the same approach in validating our framework as data-driven methods and capture the training loss for different privacy budgets and non-private learning. Experiment 4 evaluates how well our adaptive budgeting scheme maintains convergence when we vary ϵ, and Experiment 5 assesses the influence of the other parameters on the learning utility.

Experiment 4 (RQ2.2) Assesses how well our adaptive budgeting scheme maintains convergence with variable privacy budgets.

Setup: We keep all the parameters to the same values and vary the privacy budget ϵ. We train the model with AdaVFL until the budget is exhausted and use the same number of epochs to train a new model without privacy.

Results: Figures 6.13c and 6.13d show that for $\epsilon = 1.6$, the convergence speed of AdaVFL is slower than non-private learning for the two datasets, and the training loss settles at a slightly higher value. When we increase ϵ to 1.8, we notice that the convergence improves (Figures 6.13a and 6.13b). This observation is consistent with Theorem 2 in K. Wei et al. (2020), as a larger ϵ leads to a tighter bound. However, when we decrease ϵ to 1.4, the training loss training with high noise causes the network weights to oscillate around the optimal solution (Figures 6.13e and 6.13f). This result is explained by our feedback loop that monitors the model's performance and adjusts the noise range by increasing or decreasing it to steer the noise toward convergence.

Experiment 5 (RQ2.2) Assesses the effect of regulating the weight of the convergence state and the budget update frequency on the dynamic privacy budget allocation utility.

Setup: We vary the weight of the error on the computed budget, μ (Equation 28), between 0 and 1 and the frequency of the budget adjustment, and m (Equation 26), between 5 and 15, and capture the utility metrics.

Results: Since the observed results are similar for the three utility metrics, we include WMAPE only due to lack of space. We select WMAPE because it is less sensitive to outliers than RMSE since WMAPE focuses on absolute errors. We observe that the utility overall increases with the value of μ, as illustrated in Figure 6.14. Since the privacy budget adjusts to the training accuracy, which may change irregularly, the increase has some random fluctuation. Based on Equation 28, the budget value plateaus as μ gets closer to 0. Consequently, our budget allocation behaves similarly to the uniform budget solution when $\mu = 0$. We also notice that when we make the frequency of the budget adjustment, m, longer, the accuracy decreases slightly. This decrease is because the noise scale lowers slowly, negatively affecting more iterations.

6.4.4 Discussion

Resilience to Privacy Risks and Attacks (RQ1)

Differential privacy is applied in deep learning to protect the individual participants in the training data and the learned model parameters. However, there is a disagreement in the literature about the resilience of differentially private learning against privacy attacks (Boenisch, Sperl, and Böttinger (2021)). On the other hand, many works (Stock et al. (2022), Tan, Zhang, Liu, Li, and Wu (2022), and H. Phan et al. (2020)) demonstrate the efficiency of differential privacy against attacks that target the training data. This work exemplifies how adapting traditional differential privacy to the training paradigm, vertical federated learning, can achieve protection against privacy attacks.

The effect of differential privacy on learning utility (RQ2)

Differentially private deep learning adds noise during training to protect the dataset users' privacy by limiting the privacy risk associated with publishing the model parameters. However, uniform noise (Abadi et al. (2016)) results in low utility because the added noise is identical even when the model's gradients decrease during the training. Budget allocation strategies cannot deliver the same utility as the non-private models because of the gradient perturbation. However, our experiments demonstrate that AdaVFL reduces the utility difference with the non-private solution by 15% on average compared to the state-of-the-art dynamic budget allocations (Lee and Kifer (2018), Yu et al. (2019), and Gong, Feng, and Xie (2020)). This improvement is because our adaptive strategy relies on the model's training performance to adjust the noise scale.

6.4.5 Threat to Validity

We tested our feature-aware privacy budget initialization against a privacy attack that closely mimics our vertical federated learning setup. However, we must emphasize that privacy attacks are a rich research topic. Therefore, future feature and membership inference attacks may be able to infer more than we showed in our experiments.

6.5 Summary

This chapter leverages differential privacy to protect the training data and learning models in vertical federated learning for mobility forecasting. We use zCDP, a differential privacy variant, for privacy accounting to balance a tight privacy loss estimation and utility. We proposed the Adaptive differential privacy for Vertical Federated Learning (AdaVFL) protocol to allocate the privacy budget to each organization based on its features' participation in the global model and at each iteration based on its convergence state. The experimental evaluation on two real-world datasets demonstrates that AdaVFL successfully hinders privacy attacks that target feature inference by 25% and improves accuracy by 15% on average compared to the state-of-the-art allocation budget solutions. Moreover, additional experimental evaluation against the state-of-the-art membership inference attack demonstrates the efficiency of AdaVFL in reducing the attacker's attack.

6.6 Membership Inference Attack Results

This section contains the experimental results of the membership inference attack, omitted from the main text.

6.6.1 Attack Details and Metrics

We select the membership inference attack presented by Nasr et al. (2019) because its setting is the closest to ours. The presented attack considers participants and a learning collaborator in federated learning that act as honest-but-curious adversaries. However, unlike our target model, which aims to forecast mobility, the models presented in this attack are classifiers. Therefore, the samples the attacker predicts as members or not are sequences instead of single items (image/user purchase profile).

We adopt the unsupervised model network architecture where the learning collaborator is an adversary that aims to infer whether a data sample was part of a targetted organization's training dataset based on the exchanged learning parameters during the global model's training. The learning collaborator trains an attack model that uses the targeted organization's gradient vector as input features and extracts a score that predicts the membership probability of the input data. We follow

the same neural network architecture presented by Nasr et al. (2019) as illustrated in Figure 6.15 with an input layer that takes the target model's output, label, loss, and gradient. Then, the output is flattened into a vector and concatenated as input for a fully connected encoder. The encoder's output is a single score fed into a decoder to reconstruct critical features of the attack input. The decoder only plays a role during the training of the adversary model but not during the attack.

Similar to the presented attack, we measure the following metric to assess the membership inference attack's success:

- Attack accuracy: The fraction of the correct membership predictions for unknown data points over the size of the member and non-member samples.

Experiment: Assesses the effect of the adaptive budgeting scheme on the learning collaborator membership attack accuracy.

Setup: The learning collaborator captures the target organization model's parameters over several training epochs. Then, it trains the attack model using the collected parameters and a portion of the target organization's training data. We vary the observed epochs at different phases of the training process and the ratio of the attacker's training dataset that constitutes the attacker's training model. Then, we capture the attack's accuracy.

Results: Tables 6.3 and 6.4 show the membership inference attack accuracy with variable sets of captured training epochs. We notice that the accuracy decreases (by up to 13%) when the organization applies our AdaVFL method instead of not using any privacy measure. Similar to the results observed in Nasr et al. (2019), using later epochs increases the attack accuracy because the later epochs include more membership information as the model learns the outliers in such epochs.

Tables 6.5 and 6.6 show the membership inference attack accuracy with variable sets of training data ratio. As demonstrated in Nasr et al. (2019), we observe that when the attacker is trained with more member data, the accuracy of the membership inference attack improves. Moreover, we show that using our AdaVFL method reduces the attacker's accuracy (by up to 14%).

Observed epochs	Attack accuracy with no privacy	Attack accuracy with AdaVFL (↓%)
3,5,7,9	60.9%	58.6% (↓3%)
5,7,9,11	67.7%	60.3% (↓10%)
7,9,11,13	73%	64.2% (↓12%)
9,11,13,15	78.3%	70.7% (↓9%)

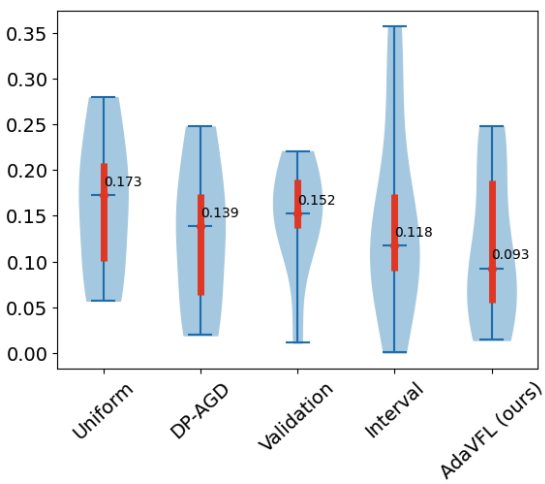
Table 6.3: A comparison of the accuracy of the learning collaborator attacker with various training epochs on Bike New York dataset. (↓%) represents the percentage decrease in the attack's accuracy compared to the no-privacy solution.

Observed epochs	Attack accuracy with no privacy	Attack accuracy with AdaVFL (↓%)
2,4,6,8	66.5%	57.8% (↓13%)
4,6,8,10	71.1%	64.6% (↓9%)
6,8,10,12	73.2%	66.4% (↓9%)
8,10,12,14	78.2%	71.5% (↓8%)

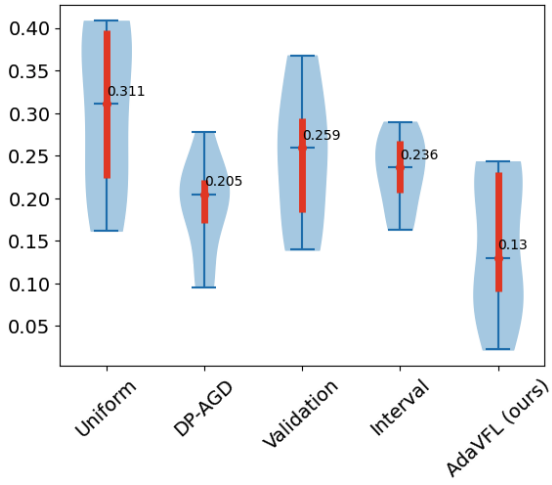
Table 6.4: A comparison of the accuracy of the learning collaborator attacker with various training epochs on Yelp review dataset. (↓%) represents the percentage decrease in the attack's accuracy compared to the no-privacy solution.

Training data member (ratio)	Training data non-member (ratio)	Attack accuracy with no privacy	Attack accuracy with AdaVFL (↓%)
20	80	66.2%	62.6%(↓5%)
30	70	73.1%	68%(↓6%)
40	60	77.6%	71.8%(↓7%)
50	50	80.8%	73.1%(↓9%)

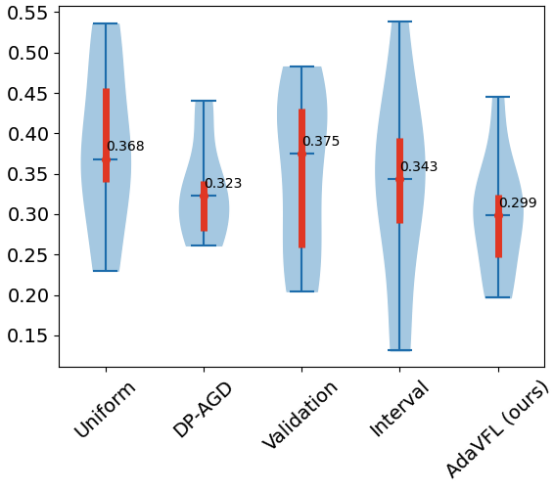
Table 6.5: A comparison of the accuracy of the learning collaborator attacker with various training data size on Bike New York dataset. (↓%) represents the percentage decrease in the attack's accuracy compared to the no-privacy solution.



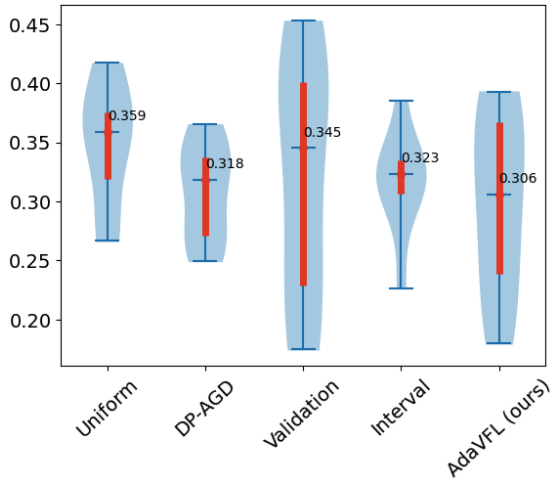
(a) Bike dataset, AE



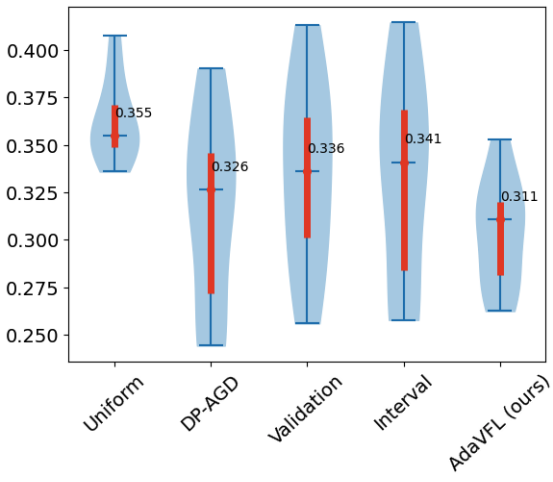
(b) Yelp reviews, AE



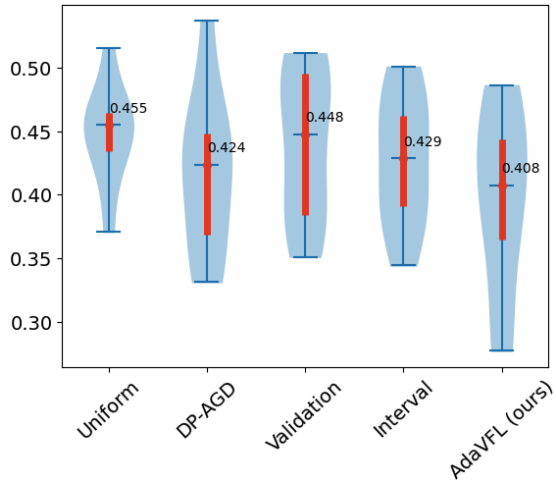
(c) Bike dataset, RMSE



(d) Yelp reviews, RMSE

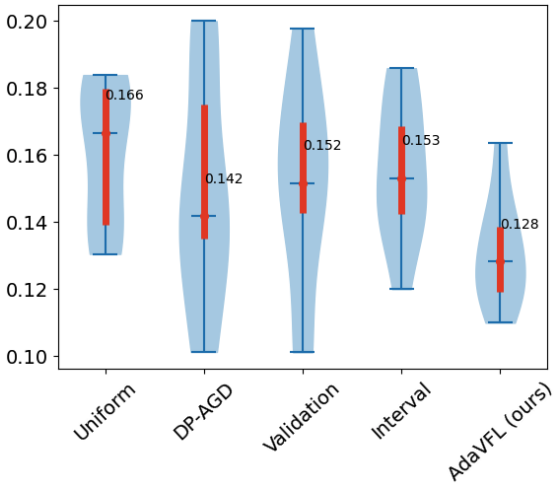


(e) Bike dataset,WMAPE

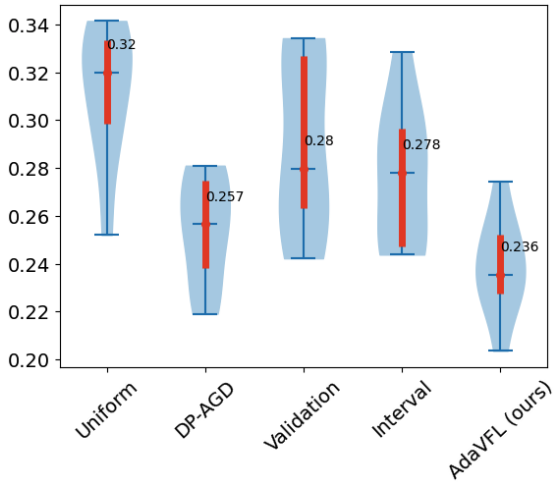


(f) Yelp reviews, WMAPE

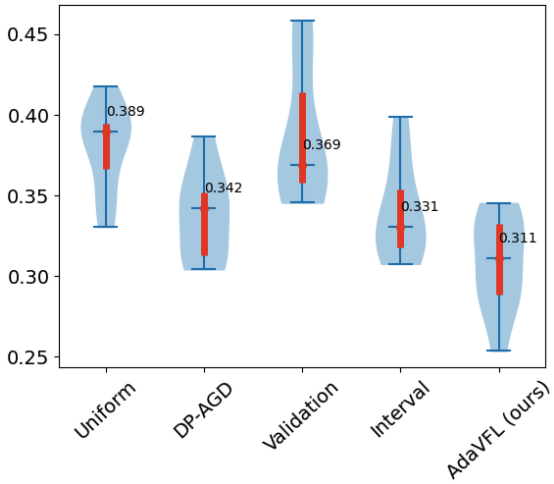
Figure 6.8: The utility of the local model under fixed training epochs on Bike New York and Yelp review datasets.



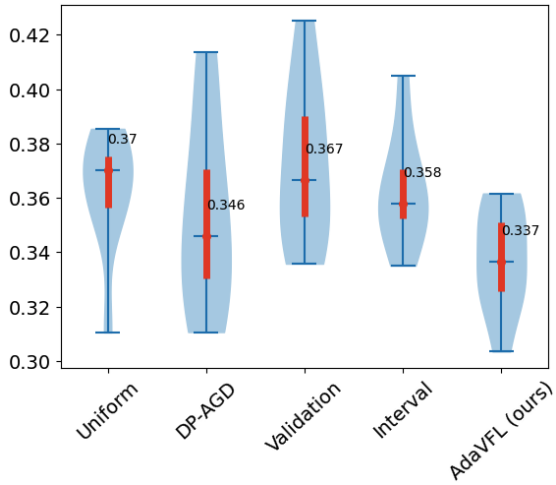
(a) Bike dataset, AE



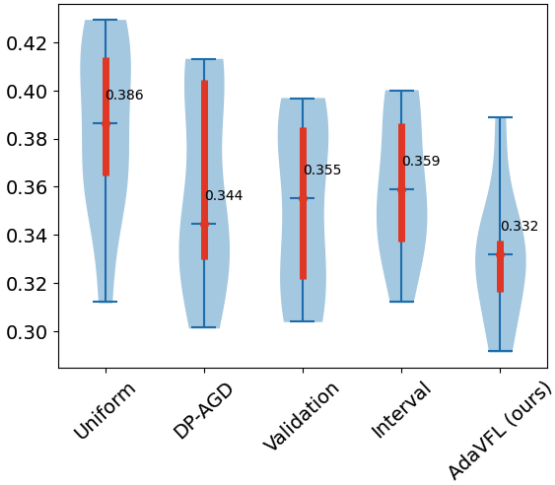
(b) Yelp reviews, AE



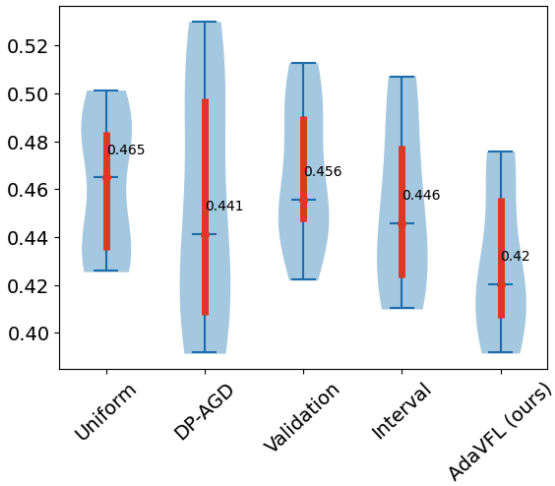
(c) Bike dataset, RMSE



(d) Yelp reviews, RMSE

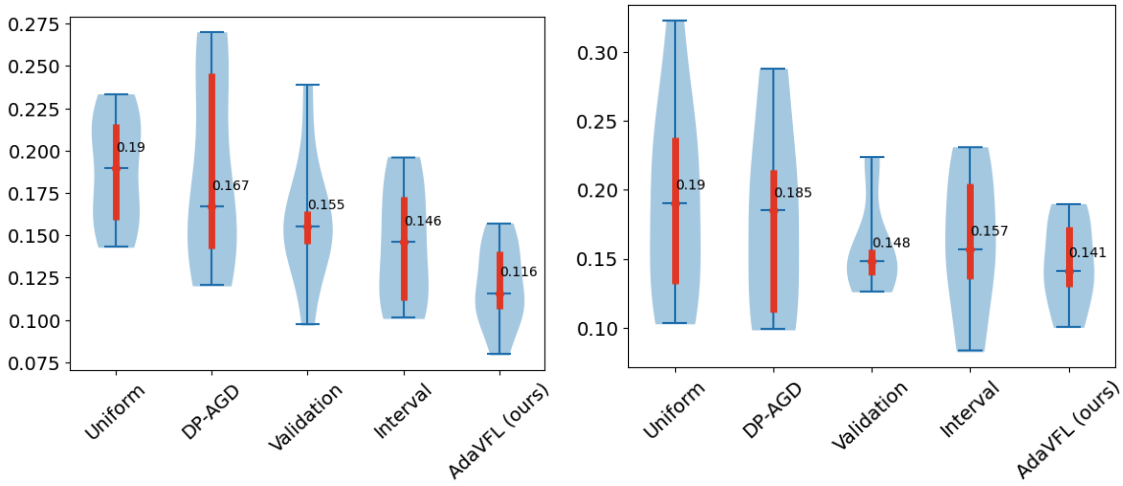


(e) Bike dataset,WMAPE



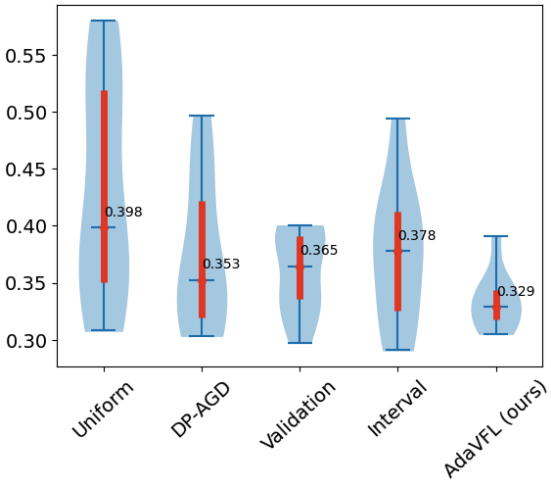
(f) Yelp reviews, WMAPE

Figure 6.9: The utility of the global model under fixed training epochs on Bike New York and Yelp review datasets.

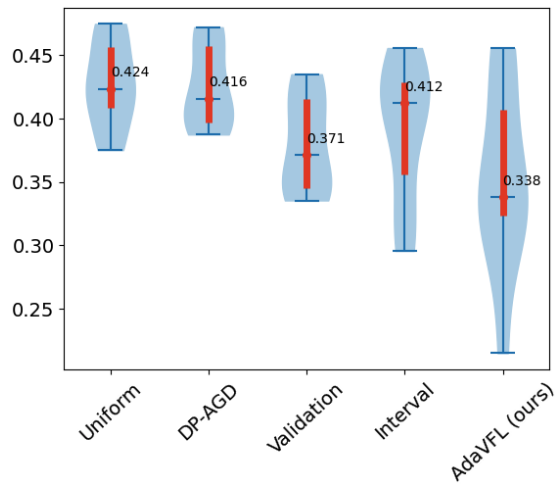


(a) Bike dataset, AE

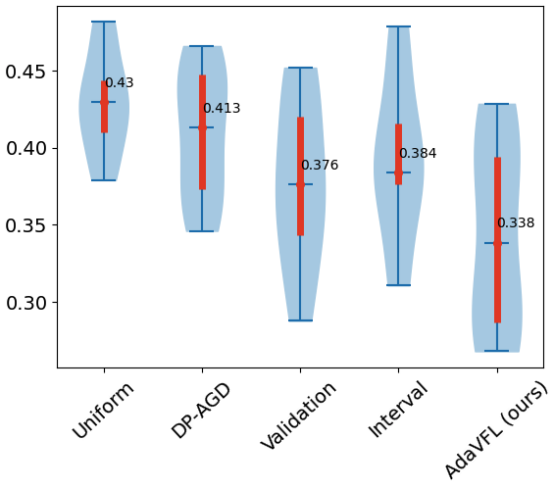
(b) Yelp reviews, AE



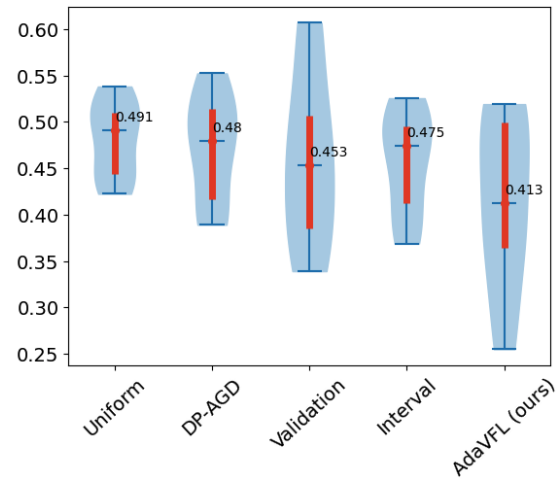
(c) Bike dataset, RMSE



(d) Yelp reviews, RMSE

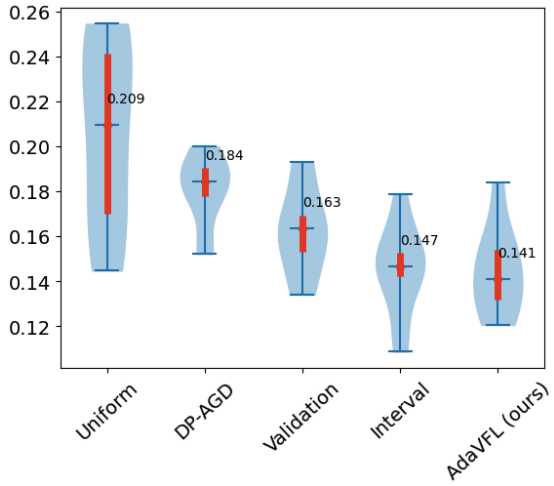


(e) Bike dataset, WMAPE

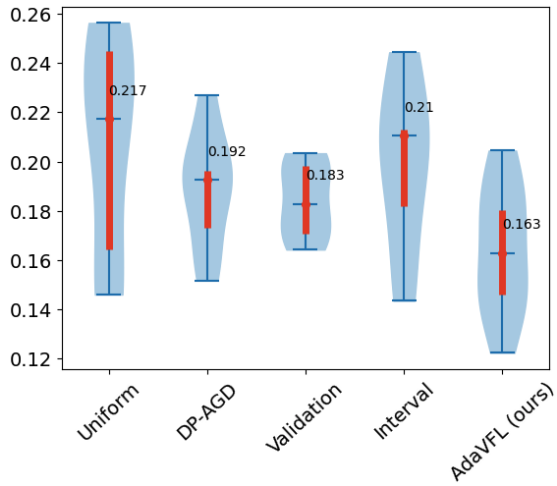


(f) Yelp reviews, WMAPE

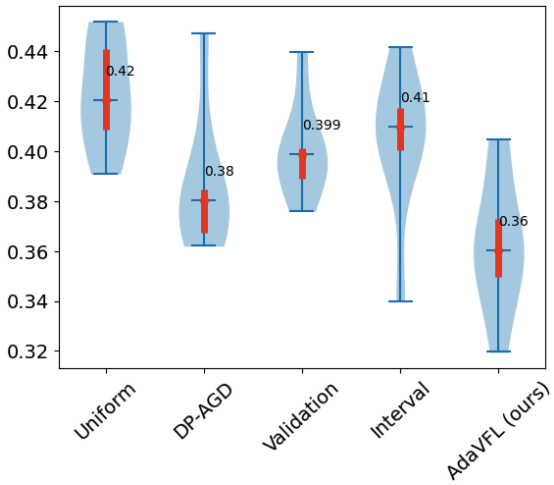
Figure 6.10: The utility of the local model under a fixed privacy budget on Bike New York and Yelp review datasets.



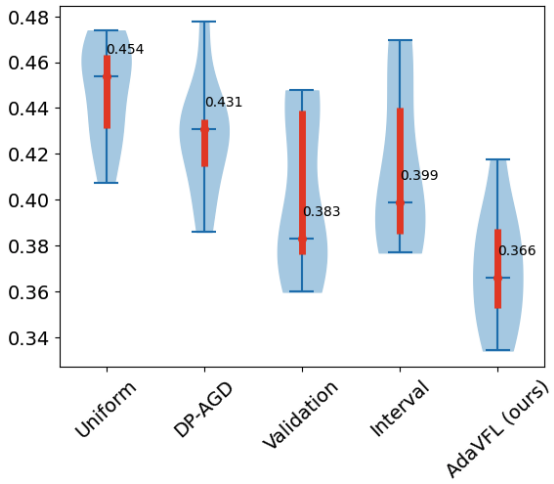
(a) Bike dataset, AE



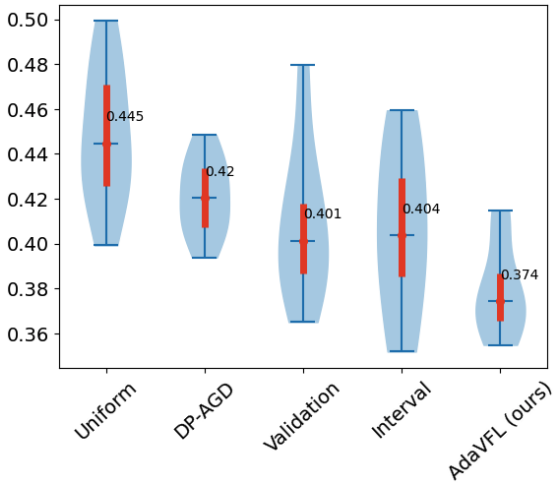
(b) Yelp reviews, AE



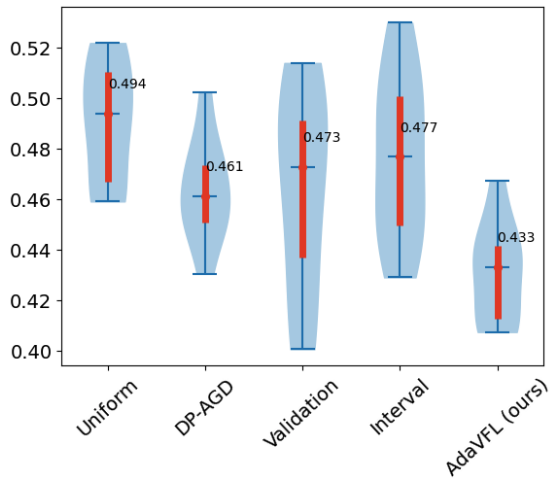
(c) Bike dataset, RMSE



(d) Yelp reviews, RMSE



(e) Bike dataset, WMAPE



(f) Yelp reviews, WMAPE

Figure 6.11: The utility of the global model under a fixed privacy budget on Bike New York and Yelp review datasets.

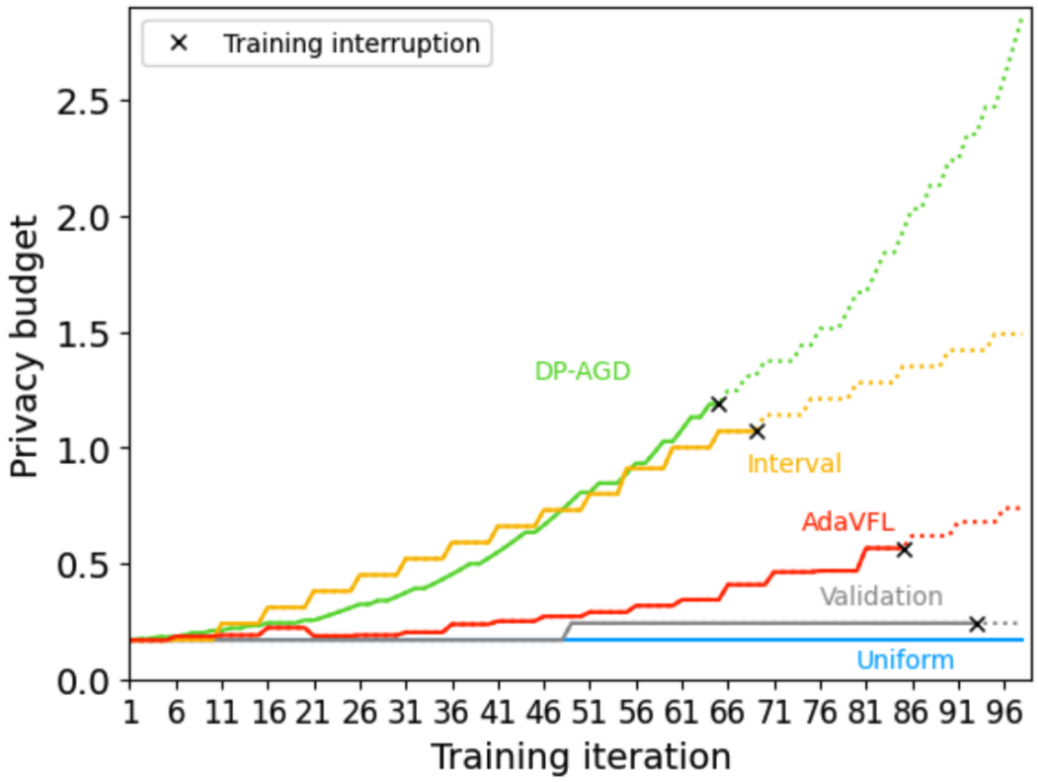
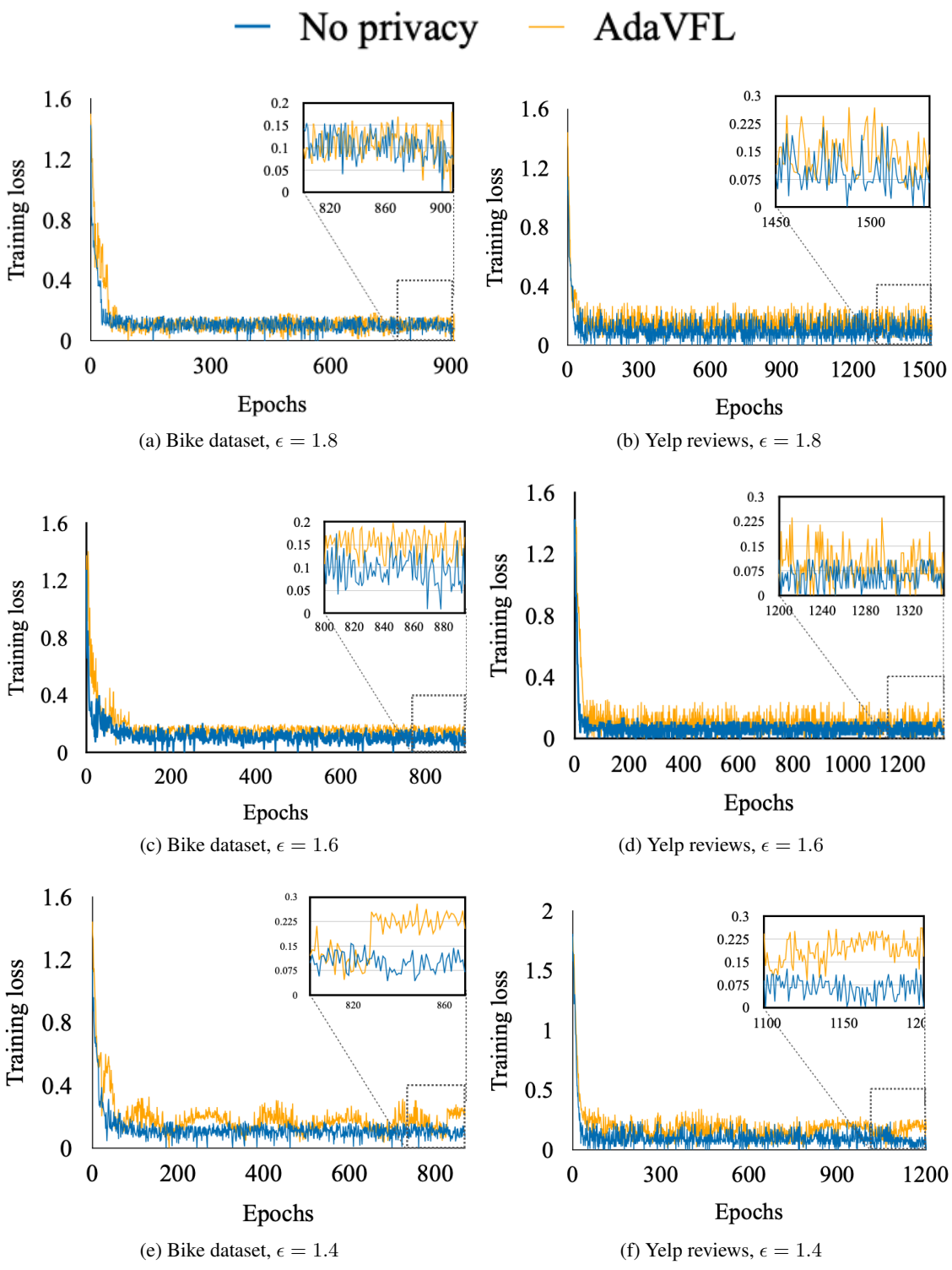


Figure 6.12: The budget assigned at each iteration per solution captured for a fixed number of iterations. The beginning of the dotted lines indicates the iteration where the overall budget is exhausted.

Training data member (ratio)	**Training data non-member (ratio)**	**Attack accuracy with no privacy**	**Attack accuracy with AdaVFL (↓%)**
20	80	60.7%	57%(↓6%)
30	70	69.8%	59.6%(↓14%)
40	60	73%	65.9% (↓%9)
50	50	77%	70.5% (↓8%)

Table 6.6: A comparison of the accuracy of the learning collaborator attacker with various training data size on Yelp review dataset. (↓%) represents the percentage decrease in the attack's accuracy compared to the no-privacy solution.



(a) Bike dataset, $\epsilon = 1.8$

(b) Yelp reviews, $\epsilon = 1.8$

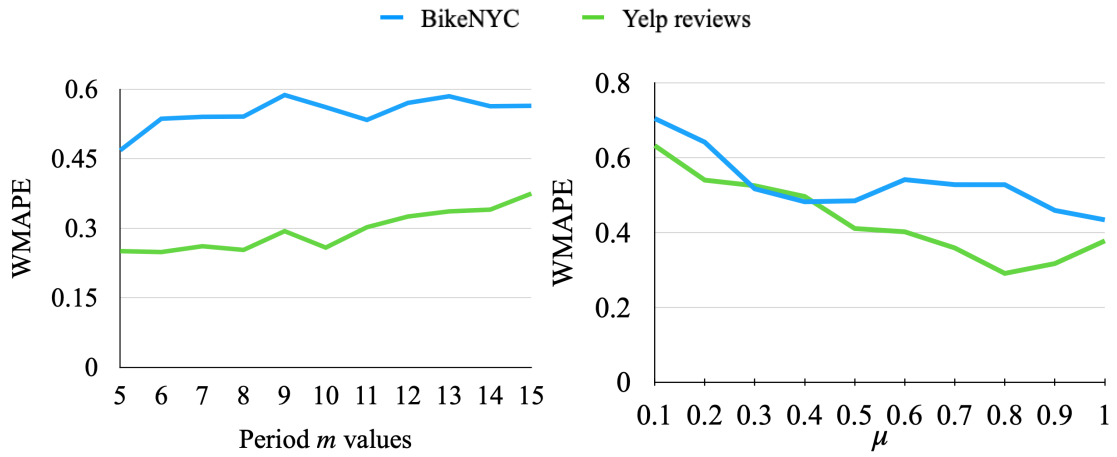
(c) Bike dataset, $\epsilon = 1.6$

(d) Yelp reviews, $\epsilon = 1.6$

(e) Bike dataset, $\epsilon = 1.4$

(f) Yelp reviews, $\epsilon = 1.4$

Figure 6.13: Convergence results of with AdaVFL with different noise levels ($\epsilon = 1.4$, $\epsilon = 1.6$, $\epsilon = 1.8$) and without noise on Bike New York and Yelp reviews datasets.



(a) Different m values

(b) Different μ values

Figure 6.14: The effect of the algorithm parameters on utility. When mu is closer to 0, the budget allocation becomes less adaptive and behaves like the uniform budget scheme, leading to lower utility. When m is larger, it takes longer intervals to adapt the privacy budget, resulting in a slight decrease in utility.

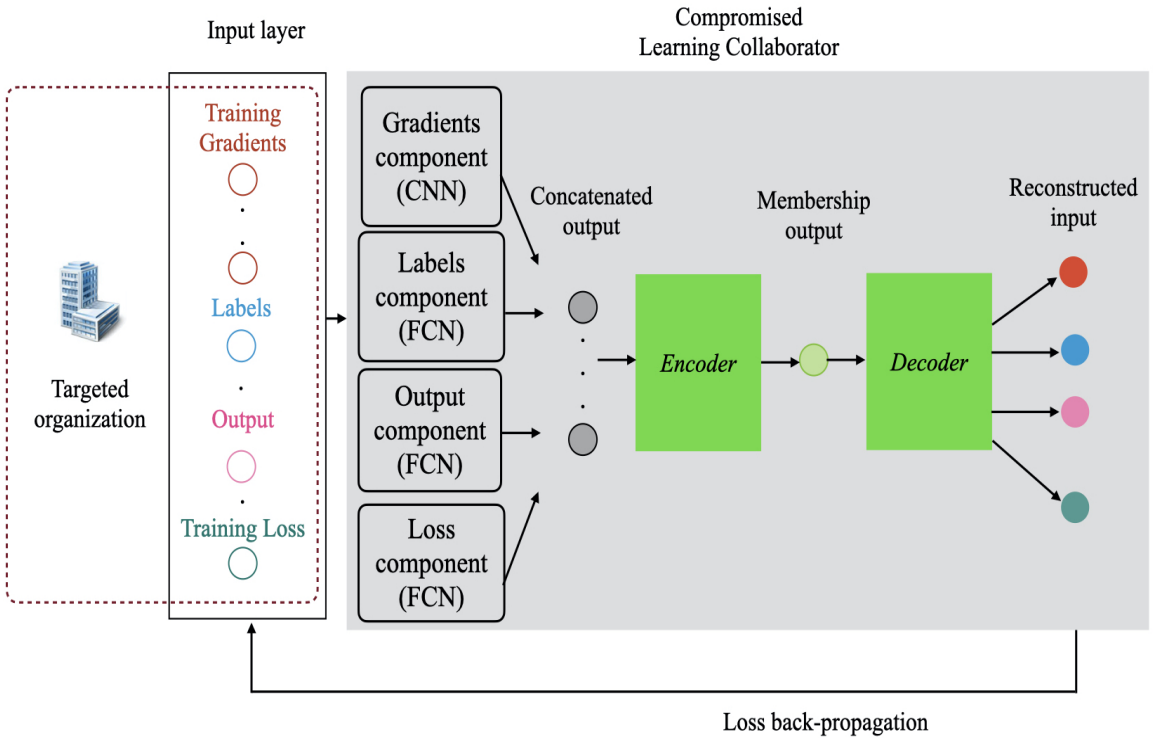


Figure 6.15: The membership inference attack details: The learning collaborator receives the intermediate training parameters from the targeted organization and uses them as input for an unsupervised attack model. The model is composed of a convolutional neural network and several fully connected networks, an encoder, and a decoder.

Chapter 7

Conclusion and Future Directions

Global positioning systems, wireless sensor networks and social media generate daily mobility data covering billions of people across broad geo-areas. Service application providers collect, store, analyze, and share these data with the public. Unsurprisingly, major corporations such as Google, Apple, Microsoft, and Facebook are interested in user mobility because of the immense business opportunities. However, mobility data sharing and analysis pose privacy threats that may lead to stalking, theft, or abduction. This thesis tackles the issue of exploring mobility data while preserving the privacy of individuals using two scenarios: continuous mobility aggregate sharing and mobility forecasting.

We select differential privacy as the primary technique for aggregating useful mobility information and forecasting while preserving individuals' privacy. Two mobility data characteristics result in challenges when applying differential privacy. First, mobility data is usually spread across multiple organizations, whereas standard differential privacy relies on a centralized trusted curator. Secondly, mobility data is typically sequential, while the guarantee provided by differential privacy degrades with consecutive aggregating of the sensitive data.

Chapter 4 presents a distributed solution enabling multiple organizations to participate in real-time mobility aggregate monitoring while protecting their users' privacy. We leverage local differential privacy, a variant of the traditional setting, for each organization to perturb its aggregates locally before sending them to the real-time monitoring entity. We define a dynamic privacy budget allocation to cater to the sequentiality of mobility aggregates. Finally, we design an approximation

strategy to reduce the perturbation error and improve the private aggregates' utility.

Next, we propose in Chapter 5 a vertical federated learning framework for mobility data forecasting to allow the learning process to be jointly conducted over multiple organizations targetting a common location domain. This step is necessary before tackling the private mobility forecasting issue. Each organization extracts the embedded spatio-temporal correlation between its locations using a local learning model. A global model synchronizes with the local models to incorporate the correlation between all the organizations' locations.

Finally, Chapter 6 leverages differential privacy to protect the training data and models in vertical federated learning for mobility forecasting. We use zCDP, a differential privacy variant, for privacy accounting to balance a tight privacy loss estimation and utility. We proposed the Adaptive differential privacy for Vertical Federated Learning (AdaVFL) protocol to allocate the privacy budget to each organization based on its features' participation in the global model and at each iteration based on its convergence state.

Privacy preservation comes at a utility cost that is even higher with distributed differential privacy, as stated by Kairouz et al. (2021), Kairouz, Oh, and Viswanath (2014), and Ye and Barg (2018). Although we presented several budget allocation approaches to counteract this issue, there are still some research gaps that we intend to address in the future. First, we believe that the dynamic allocation budget for mobility aggregate sharing can be used for aggregates, such as counts or averages. However, we would like to investigate monitoring sophisticated statistics, such as range queries, with little utility loss. Moreover, we want to improve the performance of the exponential approximation strategy execution time.

Moreover, we assume that communication between organizations is secure in all our solutions, whereas, in practice, the adversary may target private data. For instance, the attacker may be motivated to steal the mobility forecasting model (Tramèr, Zhang, Juels, Reiter, and Ristenpart (2016)) by capturing the exchanged parameters. In this case, a combination of privacy and security measures is necessary to prevent violating individuals privacy, such as the solution presented by Mohammed, Alhadidi, Fung, and Debbabi (2013).

Additionally, we assume that the communication between organizations is reliable, leading to private model updates. However, differences in communication reliability capabilities are inevitable

among organizations. Consequently, organizations could be offline during training due to unreliable networks or other factors. Therefore, we plan to investigate the integration of asynchronous federated learning strategies.

Finally, in most real situations, the organizations may cover irregular mobility periods, leading to a disparity in data spread (non-independent and non-identically distributed data). This imbalance creates challenges for practical model training. Several attempts at solving this issue are proposed by Y. Chen, Ning, Slawski, and Rangwala (2020), Nguyen et al. (2022), R. Wang, Fung, Zhu, and Peng (2021), and Zhao et al. (2018). Therefore, another research direction could be to consider the impact of this setting on individuals' privacy.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318.

Agarwal, N., Suresh, A. T., Yu, F. X. X., Kumar, S., & McMahan, B. (2018). cpSGD: Communication-efficient and differentially-private distributed sgd. *Advances in Neural Information Processing Systems*, *31*.

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., & Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 961–971.

Anzengruber, B., Pianini, D., Nieminen, J., & Ferscha, A. (2013). Predicting social density in mass events to prevent crowd disasters. *Social Informatics: 5th International Conference, SocInfo 2013, Kyoto, Japan, November 25-27, 2013, Proceedings 5*, 206–215.

Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al. (2017). Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, *13*(5), 1333–1345.

Bassily, R., Smith, A., & Thakurta, A. (2014). Private empirical risk minimization: Efficient algorithms and tight error bounds. *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, 464–473.

Bellman, R. (1964). Control theory. *Scientific American*, *211*(3), 186–201.

Bernau, D., Robl, J., Grassal, P. W., Schneider, S., & Kerschbaum, F. (2021). Comparing local and central differential privacy using membership inference attacks. *Data and Applications*

Security and Privacy XXXV: 35th Annual IFIP WG 11.3 Conference, DBSec 2021, Calgary, Canada, July 19–20, 2021, Proceedings, 22–42.

Bhagoji, A. N., Chakraborty, S., Mittal, P., & Calo, S. (2019). Analyzing federated learning through an adversarial lens. *International Conference on Machine Learning*, 634–643.

Blum, A., Ligett, K., & Roth, A. (2013). A learning theory approach to noninteractive database privacy. *J. ACM*, *60*, 12:1–12:25.

Boenisch, F., Sperl, P., & Böttinger, K. (2021). Gradient masking and the underestimated robustness threats of differential privacy in deep learning. *arXiv preprint arXiv:2105.07985*.

Bun, M., & Steinke, T. (2016). Concentrated differential privacy: Simplifications, extensions, and lower bounds. *Theory of Cryptography Conference*, 635–658.

Cao, Y., & Yoshikawa, M. (2015). Differentially private real-time data release over infinite trajectory streams. *2015 16th IEEE International Conference on Mobile Data Management*, *2*, 68-73.

Capital bike share. (2016). Available at https://www.capitalbikeshare.com/system-data.

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., & Song, D. (2019). The secret sharer: Evaluating and testing unintended memorization in neural networks. *28th USENIX Security Symposium (USENIX Security 19)*, 267–284.

Chatzikokolakis, K., Palamidessi, C., & Stronati, M. (2014). A predictive differentially-private mechanism for mobility traces. *International Symposium on Privacy Enhancing Technologies Symposium*, 21–41.

Chatzikokolakis, K., Palamidessi, C., & Stronati, M. (2015). Location privacy via geo-indistinguishability. *ACM SIGLOG News*, *2*(3), 46–69.

Chaudhuri, K., Monteleoni, C., & Sarwate, A. D. (2011). Differentially private empirical risk minimization. *Journal of Machine Learning Research*, *12*(3).

Chen, C., Lyu, L., Yu, H., & Chen, G. (2022). Practical attribute reconstruction attack against federated learning. *IEEE Transactions on Big Data*.

Chen, M., & Ohrimenko, O. (2023). Protecting global properties of datasets with distribution privacy mechanisms. *International Conference on Artificial Intelligence and Statistics*, 7472–7491.

Chen, T., Jin, X., Sun, Y., & Yin, W. (2020). Vafl: A method of vertical asynchronous federated learningg. *arXiv preprint arXiv:2007.06081*.

Chen, Y., Ning, Y., Slawski, M., & Rangwala, H. (2020). Asynchronous online federated learning for edge devices with non-iid data. *2020 IEEE International Conference on Big Data (Big Data)*, 15–24.

Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., & Yang, Q. (2021). Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, *36*(6), 87–98.

Cho, E., Myers, S. A., & Leskovec, J. (2011). Friendship and mobility: User movement in location-based social networks. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1082–1090.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Cormode, G., Procopiuc, C., Srivastava, D., Shen, E., & Yu, T. (2012). Differentially private spatial decompositions. *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, 20–31.

Dean, J., & Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113.

Ding, B., Kulkarni, J., & Yekhanin, S. (2017). Collecting telemetry data privately. *Advances in Neural Information Processing Systems*, 3571–3580.

Duchi, J. C., Jordan, M. I., & Wainwright, M. J. (2013). Local privacy and statistical minimax rates. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 429–438.

Dwork, C. (2008). Differential privacy: A survey of results. *International Conference on Theory and Applications of Models of Computation*, 1–19.

Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, 265–284.

Dwork, C., Naor, M., Pitassi, T., & Rothblum, G. N. (2010). Differential privacy under continual observation. *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, 715–724.

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

Dwork, C., Rothblum, G. N., & Vadhan, S. (2010). Boosting and differential privacy. *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 51–60.

Erlingsson, Ú., Pihur, V., & Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 1054–1067.

Errounda, F. Z., & Liu, Y. (2018). Continuous location statistics sharing algorithm with local differential privacy. *2018 IEEE International Conference on Big Data (Big Data)*, 5147-5152.

Errounda, F. Z., & Liu, Y. (2019). An analysis of differential privacy research in location data. *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, 53–60.

Errounda, F. Z., & Liu, Y. (2021). Collective location statistics release with local differential privacy. *Future Generation Computer Systems*, *124*, 174–186.

Errounda, F. Z., & Liu, Y. (2022). A mobility forecasting framework with vertical federated learning. *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, 301–310.

Errounda, F. Z., & Liu, Y. (2023). Adaptive differential privacy in vertical federated learning for mobility forecasting. *Future Generation Computer Systems*, *149*, 531–546.

Fan, L., & Xiong, L. (2014). An adaptive approach to real-time aggregate monitoring with differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, *26*(9), 2094–2106.

Fan, L., Xiong, L., & Sunderam, V. (2013). Fast: Differentially private real-time aggregate monitor with filtering and adaptive sampling. *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 1065–1068.

Farokhi, F., & Kaafar, M. A. (2020). Modelling and quantifying membership information leakage in machine learning. *arXiv preprint arXiv:2001.10648*.

Feng, J., Rong, C., Sun, F., Guo, D., & Li, Y. (2020). Pmf: A privacy-preserving human mobility

prediction framework via federated learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, *4*(1), 1–21.

Fredrikson, M., Jha, S., & Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 1322–1333.

Frigerio, L., de Oliveira, A. S., Gomez, L., & Duverger, P. (2019). Differentially private generative adversarial networks for time series, continuous, and discrete open data. *ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings 34*, 151–164.

Fu, C., Zhang, X., Ji, S., Chen, J., Wu, J., Guo, S., ... Wang, T. (2022). Label inference attacks against vertical federated learning. *31st USENIX Security Symposium (USENIX Security 22), Boston, MA*.

Fu, F., Shao, Y., Yu, L., Jiang, J., Xue, H., Tao, Y., & Cui, B. (2021). Vf2boost: Very fast vertical federated gradient boosting for cross-enterprise learning. *Proceedings of the 2021 International Conference on Management of Data*, 563–576.

Gambs, S., Killijian, M.-O., & del Prado Cortez, M. N. (2010). Show me how you move and i will tell you who you are. *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, 34–41.

Ganaie, M. A., Hu, M., Malik, A., Tanveer, M., & Suganthan, P. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, *115*, 105151.

Geng, Q., & Viswanath, P. (2016). The optimal noise-adding mechanism in differential privacy. *IEEE Transactions on Information Theory*, *62*(2), 925–951.

Gentry, C. (2010). Computing arbitrary functions of encrypted data. *Communications of the ACM*, *53*(3), 97–105.

Georgiadou, Y., de By, R. A., & Kounadi, O. (2019). Location privacy in the wake of the gdpr. *ISPRS International Journal of Geo-Information*, *8*(3), 157.

Ghazi, B., Golowich, N., Kumar, R., Manurangsi, P., & Zhang, C. (2021). Deep learning with label differential privacy. *Advances in Neural Information Processing Systems*, *34*, 27131–27145.

Gong, M., Feng, J., & Xie, Y. (2020). Privacy-enhanced multi-party deep learning. *Neural Networks*, *121*, 484–496.

Gong, M., Pan, K., Xie, Y., Qin, A. K., & Tang, Z. (2020). Preserving differential privacy in deep neural networks with relevance-based adaptive noise imposition. *Neural Networks*, *125*, 131–141.

Gordon-Koven, L., & Levenson, N. (2014). Citi bike takes new york. *Rudin Center for Transportation Management and Policy*.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232.

Gruteser, M., & Grunwald, D. (2003). Anonymous usage of location-based services through spatial and temporal cloaking. *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, 31–42.

Guo, P., Ye, B., Chen, Y., Li, T., Yang, Y., & Qian, X. (2021). A location data protection protocol based on differential privacy. *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 306–311.

Guo, S., Lin, Y., Li, S., Chen, Z., & Wan, H. (2019). Deep spatial–temporal 3d convolutional neural networks for traffic data forecasting. *IEEE Transactions on Intelligent Transportation Systems*, *20*(10), 3913–3926.

Gupta, O., & Raskar, R. (2018). Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, *116*, 1–8.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, 47–57.

Han, X., Ding, R., Wang, L., & Huang, H. (2019). Creditprint: Credit investigation via geographic footprints by deep learning. *arXiv preprint arXiv:1910.08734*.

Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., ... Ramage, D. (2018). Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*.

Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., & Thorne, B. (2017). Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*.

Hilbert, D. (1935). Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter band: Analysis· grundlagen der mathematik· physik verschiedenes* (pp. 1–2). Springer.

Hitaj, B., Ateniese, G., & Perez-Cruz, F. (2017). Deep models under the gan: Information leakage from collaborative deep learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 603–618.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hong, J., Wang, Z., & Zhou, J. (2022). Dynamic privacy budget allocation improves data efficiency of differentially private gradient descent. *2022 ACM Conference on Fairness, Accountability, and Transparency*, 11–35.

Hu, R., Guo, Y., & Gong, Y. (2021). Concentrated differentially private federated learning with performance analysis. *IEEE Open Journal of the Computer Society*, *2*, 276–289.

Hua, J., Tong, W., Xu, F., & Zhong, S. (2018). A geo-indistinguishable location perturbation mechanism for location-based services supporting frequent queries. *IEEE Transactions on Information Forensics and Security*, *13*(5), 1155–1168.

Huang, Z., Hu, R., Guo, Y., Chan-Tin, E., & Gong, Y. (2019). Dp-admm: Admm-based distributed learning with differential privacy. *IEEE Transactions on Information Forensics and Security*, *15*, 1002–1012.

Ilin, C., Annan-Phan, S., Tai, X. H., Mehra, S., Hsiang, S., & Blumenstock, J. E. (2021). Public mobility data enables covid-19 forecasting and management at local and global scales. *Scientific reports*, *11*(1), 1–11.

Islam, M. S., Kuzu, M., & Kantarcioglu, M. (2012). Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. *Network and Distributed System Security (NDSS) Symposium*, *20*, 12.

Jo, G., Jung, K., & Park, S. (2018). An adaptive window size selection method for differentially private data publishing over infinite trajectory stream. *Journal of Advanced Transportation*,

2018.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... others (2021). Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, *14*(1–2), 1–210.

Kairouz, P., Oh, S., & Viswanath, P. (2014). Extremal mechanisms for local differential privacy. *Advances in Neural Information Processing Systems*, *27*.

Kellaris, G., Papadopoulos, S., Xiao, X., & Papadias, D. (2014). Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, *7*(12), 1155–1166.

Khan, A., Baharudin, B., Lee, L. H., & Khan, K. (2010). A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology*, *1*(1), 4–20.

Kim, J. W., Kim, D.-H., & Jang, B. (2018). Application of local differential privacy to collection of indoor positioning data. *IEEE Access*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Konečnỳ, J., McMahan, H. B., Ramage, D., & Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Kothai, G., Poovammal, E., Dhiman, G., Ramana, K., Sharma, A., AlZain, M. A., ... Masud, M. (2021). A new hybrid deep learning algorithm for prediction of wide traffic congestion in smart cities. *Wireless Communications and Mobile Computing*, *2021*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1097–1105.

Krumm, J. (2007). Inference attacks on location tracks. *International Conference on Pervasive Computing*, 127–143.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, *8*(1), 98–113.

Le, N. K., Liu, Y., Nguyen, Q. M., Liu, Q., Liu, F., Cai, Q., & Hirche, S. (2021). Fedxgboost:

Privacy-preserving xgboost for federated learning. *arXiv preprint arXiv:2106.10662*.

Lee, J., & Kifer, D. (2018). Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1656–1665.

Li, H., Xiong, L., Jiang, X., & Liu, J. (2015). Differentially private histogram publication for dynamic datasets: An adaptive sampling approach. *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 1001–1010.

Li, N., Yang, W., & Qardaji, W. (2013). Differentially private grids for geospatial data. *Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013)*, 757–768.

Li, Q., Cao, G., & La Porta, T. F. (2014). Efficient and privacy-aware data aggregation in mobile sensing. *IEEE Transactions on Dependable and Secure Computing*, *11*(2), 115–129.

Li, W., Tao, W., Qiu, J., Liu, X., Zhou, X., & Pan, Z. (2019). Densely connected convolutional networks with attention lstm for crowd flows prediction. *IEEE Access*, *7*, 140488–140498.

Liang, W., Chen, H., Wu, Y., & Li, C. (2019). Differentially private frequent itemset mining against incremental updates. *International Conference on Information and Communications Security*, 649–667.

Lin, H., Liu, G., Li, F., & Zuo, Y. (2021). Where to go? predicting next location in iot environment. *Frontiers of Computer Science*, *15*(1), 1–13.

Liu, B., Ding, M., Shaham, S., Rahayu, W., Farokhi, F., & Lin, Z. (2021). When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)*, *54*(2), 1–36.

Liu, B., Zhou, W., Zhu, T., Gao, L., & Xiang, Y. (2018). Location privacy and its applications: A systematic study. *IEEE access*, *6*, 17606–17624.

Liu, G., Yin, Z., Jia, Y., & Xie, Y. (2017). Passenger flow estimation based on convolutional neural network in public transportation system. *Knowledge-Based Systems*, *123*, 102–115.

Liu, H., Li, C., Liu, B., Wang, P., Ge, S., & Wang, W. (2021). Differentially private learning with grouped gradient clipping. In *Acm multimedia asia* (pp. 1–7).

Liu, L., Biderman, A., & Ratti, C. (2009). Urban mobility landscape: Real time monitoring of urban mobility patterns. *Proceedings of the 11th International Conference on Computers in Urban Planning and Urban Management*, 1–16.

Liu, Y., James, J., Kang, J., Niyato, D., & Zhang, S. (2020). Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet of Things Journal*, *7*(8), 7751–7763.

Liu, Y., Zhang, S., Zhang, C., & James, J. (2020). Fedgru: Privacy-preserving traffic flow prediction via federated learning. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6.

Lonare, S., & Bhramaramba, R. (2021). Model aggregation federated learning approach for vehicular traffic forecasting. *Journal of Engineering Science & Technology Review*, *14*(3).

Lu, R., Liang, X., Li, X., Lin, X., & Shen, X. (2012). Eppa: An efficient and privacy-preserving aggregation scheme for secure smart grid communications. *IEEE Transactions on Parallel and Distributed Systems*, *23*(9), 1621–1631.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*.

Luo, X., Wu, Y., Xiao, X., & Ooi, B. C. (2021). Feature inference attack on model predictions in vertical federated learning. *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 181–192.

Ma, X., Ma, J., Li, H., Jiang, Q., & Gao, S. (2017). Agent: An adaptive geo-indistinguishable mechanism for continuous location-based service. *Peer-to-Peer Networking and Applications*, 1–13.

Manh, H., & Alaghband, G. (2018). Scene-lstm: A model for human trajectory prediction. *arXiv preprint arXiv:1808.04018*.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. *Artificial Intelligence and Statistics*, 1273–1282.

McSherry, F., & Talwar, K. (2007). Mechanism design via differential privacy. *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, 94–103.

Mohammed, N., Alhadidi, D., Fung, B. C., & Debbabi, M. (2013). Secure two-party differentially private data release for vertically partitioned data. *IEEE Transactions on Dependable and Secure Computing*, *11*(1), 59–71.

Montavon, G., Binder, A., Lapuschkin, S., Samek, W., & Müller, K.-R. (2019). Layer-wise relevance propagation: An overview. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 193–209.

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., & Srivastava, G. (2021). A survey on security and privacy of federated learning. *Future Generation Computer Systems*, *115*, 619–640.

Murakami, T., & Kawamoto, Y. (2018). Restricted local differential privacy for distribution estimation with high data utility. *arXiv preprint arXiv:1807.11317*.

Murtagh, J., & Vadhan, S. (2015). The complexity of computing the optimal composition of differential privacy. *Theory of Cryptography: 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, 157–175.

Nasr, M., Shokri, R., & Houmansadr, A. (2019). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *2019 IEEE Symposium on Security and Privacy (SP)*, 739–753.

Ngo, H., & Kim, J. (2015). Location privacy via differential private perturbation of cloaking area. *2015 IEEE 28th Computer Security Foundations Symposium*, 63–74.

Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M., & Huba, D. (2022). Federated learning with buffered asynchronous aggregation. *International Conference on Artificial Intelligence and Statistics*, 3581–3607.

Nie, Y., Huang, L., Li, Z., Wang, S., Zhao, Z., Yang, W., & Lu, X. (2016). Geospatial streams publish with differential privacy. *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 152–164.

Nock, R., Hardy, S., Henecka, W., Ivey-Law, H., Patrini, G., Smith, G., & Thorne, B. (2018). Entity resolution and federated learning get a federated resolution. *arXiv preprint arXiv:1803.04035*.

Ny, J. L., Touati, A., & Pappas, G. J. (2014). Real-time privacy-preserving model-based estimation of traffic flows. *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*, 92-102.

Oliver, N., Letouzé, E., Sterly, H., Delataille, S., De Nadai, M., Lepri, B., ... others (2020). Mobile

phone data and covid-19: Missing an opportunity? *arXiv preprint arXiv:2003.12347*.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*, 8026–8037.

Paverd, A., Martin, A., & Brown, I. (2014). Modelling and automatically analysing privacy properties for honest-but-curious adversaries. *Tech. Rep*.

Phan, H., Thai, M. T., Hu, H., Jin, R., Sun, T., & Dou, D. (2020). Scalable differential privacy with certified robustness in adversarial learning. *International Conference on Machine Learning*, 7683–7694.

Phan, N., Vu, M., Liu, Y., Jin, R., Dou, D., Wu, X., & Thai, M. T. (2019). Heterogeneous gaussian mechanism: Preserving differential privacy in deep learning with provable robustness. *arXiv preprint arXiv:1906.01444*.

Phan, N., Wu, X., Hu, H., & Dou, D. (2017). Adaptive laplace mechanism: Differential privacy preservation in deep learning. *2017 IEEE International Conference on Data Mining (ICDM)*, 385–394.

Pichapati, V., Suresh, A. T., Yu, F. X., Reddi, S. J., & Kumar, S. (2019). Adaclip: Adaptive clipping for private sgd. *arXiv preprint arXiv:1908.07643*.

Pyrgelis, A., Troncoso, C., & De Cristofaro, E. (2017). Knock knock, who's there? membership inference on aggregate location data. *Network and Distributed System Security (NDSS) Symposium*.

Qin, C., Schlemper, J., Caballero, J., Price, A. N., Hajnal, J. V., & Rueckert, D. (2018). Convolutional recurrent neural networks for dynamic mr image reconstruction. *IEEE Transactions on Medical Imaging*, *38*(1), 280–290.

Rao, A. S. S., & Vazquez, J. A. (2020). Identification of covid-19 can be quicker through artificial intelligence framework using a mobile phone–based survey when cities and towns are under quarantine. *Infection Control & Hospital Epidemiology*, *41*(7), 826–830.

Romanini, D., Hall, A. J., Papadopoulos, P., Titcombe, T., Ismail, A., Cebere, T., ... Hoeh, M. A.

(2021). Pyvertical: A vertical federated learning framework for multi-headed splitnn. *ICLR 2021 Workshop on Distributed and Private Machine Learning*.

Rose, G. (2006). Mobile phones as traffic probes: Practices, prospects and issues. *Transport Reviews*, *26*(3), 275–291.

Roy, P., Sarkar, S., Biswas, S., Chen, F., Chen, Z., Ramakrishnan, N., & Lu, C.-T. (2021). Deep diffusion-based forecasting of covid-19 by incorporating network-level mobility information. *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 168–175.

Sadeh, N., Hong, J., Cranor, L., Fette, I., Kelley, P., Prabaker, M., & Rao, J. (2009). Understanding and capturing peoples privacy policies in a mobile social networking application. *Personal and Ubiquitous Computing*, *13*, 401–412.

Sadilek, A., & Krumm, J. (2012). Far out: Predicting long-term human mobility. *Proceedings of the AAAI Conference on Artificial Intelligence*, *26*(1), 814–820.

Sahakyan, A. B., Chambers, V. S., Marsico, G., Santner, T., Di Antonio, M., & Balasubramanian, S. (2017). Machine learning model for sequence-driven dna g-quadruplex formation. *Scientific reports*, *7*(1), 1–11.

Salem, A. M. G., Bhattacharyya, A., Backes, M., Fritz, M., & Zhang, Y. (2020). Updates-leak: Data set inference and reconstruction attacks in online learning. *29th USENIX Security Symposium*, 1291–1308.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.

Shou, L., Shang, X., Chen, K., Chen, G., & Zhang, C. (2013). Supporting pattern-preserving anonymization for time-series data. *IEEE Transactions on Knowledge and Data Engineering*, *25*(4), 877–892.

Singh, U., Determe, J.-F., Horlin, F., & De Doncker, P. (2020). Crowd forecasting based on wifi sensors and lstm neural networks. *IEEE Transactions on Instrumentation and Measurement*, *69*(9), 6121–6131.

Song, S., Chaudhuri, K., & Sarwate, A. D. (2013). Stochastic gradient descent with differentially

private updates. *2013 IEEE Global Conference on Signal and Information Processing*, 245–248.

Song, X., Fan, X., Xiang, C., Ye, Q., Liu, L., Wang, Z., ... Fang, G. (2019). A novel convolutional neural network based indoor localization framework with wifi fingerprinting. *IEEE Access*, *7*, 110698–110709.

Stavroulakis, P., & Stamp, M. (2010). *Handbook of information and communication security*. Springer Science & Business Media.

Stock, P., Shilov, I., Mironov, I., & Sablayrolles, A. (2022). Defending against reconstruction attacks with r\'enyi differential privacy. *arXiv preprint arXiv:2202.07623*.

Sun, H., Yang, C., Deng, L., Zhou, F., Huang, F., & Zheng, K. (2021). Periodicmove: Shift-aware human mobility recovery with graph neural network. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 1734–1743.

Taïk, A., & Cherkaoui, S. (2020). Electrical load forecasting using edge computing and federated learning. *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, 1–6.

Tan, J., Zhang, L., Liu, Y., Li, A., & Wu, Y. (2022). Residue-based label protection mechanisms in vertical logistic regression. *arXiv preprint arXiv:2205.04166*.

Tang, X., Nasr, M., Mahloujifar, S., Shejwalkar, V., Song, L., Houmansadr, A., & Mittal, P. (2022). Machine learning with differentially private labels: Mechanisms and frameworks. *Proceedings on Privacy Enhancing Technologies*, *4*, 332–350.

Tao, X., Peng, Y., Zhao, F., Wang, S., & Liu, Z. (2020). An improved parallel network traffic anomaly detection method based on bagging and gru. *International Conference on Wireless Algorithms, Systems, and Applications*, 420–431.

Thompson, S. A., & Warzel, C. (2022). Twelve million phones, one dataset, zero privacy. *Ethics of Data and Analytics*, 161–169.

To, H., Ghinita, G., Fan, L., & Shahabi, C. (2017). Differentially private location protection for worker datasets in spatial crowdsourcing. *IEEE Transactions on Mobile Computing*, *16*(4), 934–949.

To, H., Ghinita, G., & Shahabi, C. (2014). A framework for protecting worker location privacy in spatial crowdsourcing. *Proceedings of the VLDB Endowment*, *7*(10), 919–930.

Tockar, A. (2014). Riding with the stars: Passenger privacy in the nyc taxicab dataset. *Neustar Research, September*, *15*, 6.

Tramer, F., & Boneh, D. (2020). Differentially private learning needs better features (or much more data). *arXiv preprint arXiv:2011.11660*.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., & Ristenpart, T. (2016). Stealing machine learning models via prediction {APIs}. *25th USENIX security symposium (USENIX Security 16)*, 601–618.

Truex, S., Liu, L., Chow, K.-H., Gursoy, M. E., & Wei, W. (2020). Ldp-fed: Federated learning with local differential privacy. *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, 61–66.

Tudor, V., Gulisano, V., Almgren, M., & Papatriantafilou, M. (2020). Bes: Differentially private event aggregation for large-scale iot-based systems. *Future Generation Computer Systems*, *108*, 1241–1257.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. *International Conference on Learning Representations*.

Wang, G., Dang, C. X., & Zhou, Z. (2019). Measure contribution of participants in federated learning. *2019 IEEE International Conference on Big Data (Big Data)*, 2597–2604.

Wang, L., Zhang, D., Yang, D., Lim, B. Y., Han, X., & Ma, X. (2020). Sparse mobile crowdsensing with differential and distortion location privacy. *IEEE Transactions on Information Forensics and Security*, *15*, 2735–2749.

Wang, L., Zhang, D., Yang, D., Lim, B. Y., & Ma, X. (2016). Differential location privacy for sparse mobile crowdsensing. *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 1257–1262.

Wang, R., Fung, B. C., Zhu, Y., & Peng, Q. (2021). Differentially private data publishing for arbitrarily partitioned data. *Information Sciences*, *553*, 247–265.

Wang, S., Cao, J., Chen, H., Peng, H., & Huang, Z. (2020). Seqst-gan: Seq2seq generative adversarial nets for multi-step urban crowd flow prediction. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, *6*(4), 1–24.

Wang, X., Ma, Y., Wang, Y., Jin, W., Wang, X., Tang, J., ... Yu, J. (2020). Traffic flow prediction

via spatial temporal graph neural network. *Proceedings of The Web Conference 2020*, 1082–1092.

Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, *60*(309), 63–69.

Wei, H., Zheng, G., Yao, H., & Li, Z. (2018). Intellilight: A reinforcement learning approach for intelligent traffic light control. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2496–2505.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., … Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, *15*, 3454–3469.

Wernke, M., Skvortsov, P., Dürr, F., & Rothermel, K. (2014). A classification of location privacy attacks and approaches. *Personal and Ubiquitous Computing*, *18*(1), 163–175.

Wu, X., Li, F., Kumar, A., Chaudhuri, K., Jha, S., & Naughton, J. (2017). Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. *Proceedings of the 2017 ACM International Conference on Management of Data*, 1307–1322.

Wu, X., Zhu, X., Wu, G.-Q., & Ding, W. (2013). Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, *26*(1), 97–107.

Wu, Y., Cai, S., Xiao, X., Chen, G., & Ooi, B. C. (2020). Privacy preserving vertical federated learning for tree-based models. *arXiv preprint arXiv:2008.06170*.

Xiong, W., & Lagerström, R. (2019). Threat modeling–a systematic literature review. *Computers & security*, *84*, 53–69.

Xu, D., Yuan, S., & Wu, X. (2021). Achieving differential privacy in vertically partitioned multi-party learning. *2021 IEEE International Conference on Big Data (Big Data)*, 5474–5483.

Xu, F., Tu, Z., Li, Y., Zhang, P., Fu, X., & Jin, D. (2017). Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data. *Proceedings of the 26th International Conference on World Wide Web*, 1241–1250.

Xu, R., Baracaldo, N., Zhou, Y., Anwar, A., Joshi, J., & Ludwig, H. (2021). Fedv: Privacy-preserving federated learning over vertically partitioned data. *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, 181–192.

Xue, H., Huynh, D. Q., & Reynolds, M. (2018). Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1186–1194.

Yan, Y., Zhang, L., Sheng, Q. Z., Wang, B., Gao, X., & Cong, Y. (2019). Dynamic release of big location data based on adaptive sampling and differential privacy. *IEEE Access*, *7*, 164962–164974.

Yang, Q., Chen, Y., Guizani, M., & Lee, G. M. (2021). Spatiotemporal location differential privacy for sparse mobile crowdsensing. *2021 International Wireless Communications and Mobile Computing (IWCMC)*, 1734–1741.

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(2), 1–19.

Yang, X., Gao, L., Zheng, J., & Wei, W. (2020). Location privacy preservation mechanism for location-based service with incomplete location data. *IEEE Access*, *8*, 95843–95854.

Yang, X., Ren, X., Lin, J., & Yu, W. (2016). On binary decomposition based privacy-preserving aggregation schemes in real-time monitoring systems. *IEEE Transactions on Parallel and Distributed Systems*, *27*(10), 2967–2983.

Yang, X., Wang, T., Ren, X., & Yu, W. (2017). Survey on improving data utility in differentially private sequential data publishing. *IEEE Transactions on Big Data*.

Yao, A. C.-C. (1986). How to generate and exchange secrets. *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, 162–167.

Yao, H., Tang, X., Wei, H., Zheng, G., & Li, Z. (2019). Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 5668–5675.

Ye, M., & Barg, A. (2018). Optimal schemes for discrete distribution estimation under locally differential privacy. *IEEE Transactions on Information Theory*, *64*(8), 5662–5676.

Yelp. (2019). *Yelp public dataset.* Retrieved from https://www.yelp.com/dataset

Yin, C., Xi, J., Sun, R., & Wang, J. (2017). Location privacy protection based on differential privacy strategy for big data in industrial internet-of-things. *IEEE Transactions on Industrial Informatics*, *14*(8), 3628–3636.

Yu, L., Liu, L., Pu, C., Gursoy, M. E., & Truex, S. (2019). Differentially private model publishing for deep learning. *2019 IEEE Symposium on Security and Privacy (SP)*, 332–349.

Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., & Huang, Y. (2010). T-drive: Driving directions based on taxi trajectories. *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 99–108.

Zang, H., & Bolot, J. (2011). Anonymization of location data does not work: A large-scale measurement study. *Proceedings of the 17th annual international conference on Mobile computing and networking*, 145–156.

Zhang, C., Dang, S., Shihada, B., & Alouini, M.-S. (2021). Dual attention-based federated learning for wireless traffic prediction. *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, 1–10.

Zhang, G., Zhang, A., & Zhao, P. (2020). Locmia: Membership inference attacks against aggregated location data. *IEEE Internet of Things Journal*, *7*(12), 11778–11788.

Zhang, J., Zhang, Z., Xiao, X., Yang, Y., & Winslett, M. (2012). Functional mechanism: Regression analysis under differential privacy. *Proceedings of the VLDB Endowment*(5), 1364–1375.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*.

Zheng, Y., Xie, X., Ma, W.-Y., et al. (2010). Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, *33*(2), 32–39.

Zhou, Q., Gu, J.-J., Ling, C., Li, W.-B., Zhuang, Y., & Wang, J. (2020). Exploiting multiple correlations among urban regions for crowd flow prediction. *Journal of Computer Science and Technology*, *35*, 338–352.

Zhu, B., Tang, W., Mao, X., & Yang, W. (2020). Location-based hybrid deep learning model for purchase prediction. *2020 5th International Conference on Computational Intelligence and Applications (ICCIA)*, 161–165.

Zhu, L., Liu, Z., & Han, S. (2019). Deep leakage from gradients. *Advances in Neural Information Processing Systems*, *32*.