

Lightweight privacy preservation techniques for deep learning and inference in Internet of Things

Jiang, Linshan

2022

Jiang, L. (2022). Lightweight privacy preservation techniques for deep learning and inference in Internet of Things. Doctoral thesis, Nanyang Technological University, Singapore. <https://hdl.handle.net/10356/155000>

<https://hdl.handle.net/10356/155000>

<https://doi.org/10.32657/10356/155000>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 13 Mar 2024 18:40:05 SGT

LIGHTWEIGHT PRIVACY PRESERVATION TECHNIQUES FOR DEEP LEARNING AND INFERENCE IN INTERNET OF THINGS



Jiang Linshan

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy (Ph.D.)

2022

Statement of Originality

I hereby claim that the work expressed in this thesis is the result of original research, is free of plagiarised material, and has not been submitted for a higher degree to any other University or Institution.

05/01/2022

Date

NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU



Jiang Linshan

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigation were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

05/01/2022

Date



Tan Rui

Authorship Attribution Statement

The materials of this thesis are from four research works published in the following peer-reviewed conferences and journals in which I am listed as an author.

- Part of the Chapter 2 of this thesis is published as Section 2 of the paper **Mengyao Zheng, Dixing Xu, Linshan Jiang, Chaojie Gu, Rui Tan and Peng Chen. Challenges of privacy-preserving machine learning in IoT. The First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things (AIChallengeIoT), New York, NY, 2019, co-located with ACM SenSys.**

Author contributions: I led the writing of Section 2 of the above paper with inputs from Mengyao Zheng and guidance from Prof. Rui Tan. In the Chapter 2 of this thesis, I further extend the literature review. The Chapter 2 of this thesis does not include contributions from Dixing Xu, Chaojie Gu, and Peng Chen.

- Chapter 3 is submitted as **Linshan Jiang, Qun Song, Rui Tan and Mo Li. PriMask: Cascadable and Collusion-Resilient Data Masking for Mobile Cloud Inference.** to a conference for peer review.

Author contributions: I designed the approach, implemented it, and performed all the evaluation. I also prepared the draft of the paper. Qun Song provided assistance to the implementation of the HyperNet structure and created the related illustrations. Prof. Rui Tan and Prof. Mo Li provided comments on the regular research meetings. Prof. Rui Tan revised the paper.

- Chapter 4 is published as **Linshan Jiang, Rui Tan, Xin Lou and Guosheng Lin. On Lightweight Privacy-Preserving Collaborative Learning for Internet of Things by Independent Random Projections. ACM Transactions on Internet of Things (TIOT). vol. 2, no. 11, pp 1–32. May 2021.** A preliminary version of this work is published by the same authors in **The 4th ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI), April 16-18, 2019, Montreal, Canada.**

Author contributions: I proposed the random projection approaches, wrote the codes for the experiments, and implemented the proposed approach on an embedded hardware platform. I analyzed the results and prepared the paper draft. Xin Lou assisted the experiments and implementations. Prof. Rui Tan provided the initial research direction, made comments on the research execution, and revised the paper draft. Prof. Guosheng Lin provided guidance on the design of the proposed approach and made comments on the research execution.

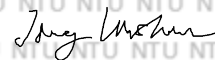
- Chapter 5 is published as **Linshan Jiang, Xin Lou, Rui Tan, and Jun Zhao. Differentially Private Collaborative Learning for the IoT Edge. The 2nd International Workshop on Crowd Intelligence for Smart Cities: Technology and Applications (CICS), Beijing, China, 2019, co-located with EWSN.**

Author contributions: I proposed the differential privacy approach and wrote the codes for experimentation. Moreover, I derived the proof, analyzed the numerical results and prepared the paper draft. Xin Lou provided the initial research direction and made suggestions on the design of experiments. Prof. Rui Tan revised the paper draft. Prof. Jun Zhao verified the derivations and made comments on the paper draft.

05/01/2022

Date

NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU



Jiang Linshan

Abstract

With the rapid development of sensing and communication technologies, the Internet of Things (IoT) is becoming a global data generation infrastructure. To utilize the massive data generated by IoT for achieving better system intelligence, machine learning and inference on the IoT data at the edge and core (i.e., cloud) of the IoT are needed. However, the pervasive data collection and processing engender various privacy concerns. While various privacy preservation mechanisms have been proposed in the context of cloud computing, they may be ill-suited for IoT due to the resource constraints at the IoT edge. This thesis primarily studies data obfuscation as a lightweight method to preserve data privacy for cloud-based collaborative machine learning and inference in IoT. Specifically, it presents three approaches: the first is for cascadable, collusion-resilient, and privacy-preserving cloud-based inference and the other two are for privacy-preserving collaborative training of a deep neural network based on distributed IoT data. All approaches protect the privacy contained in the data contributed by distributed IoT devices as the participants against a honest-but-curious coordinator in the centralized cloud system. They deliver different privacy protection properties. The first approach protects the raw forms and certain privacy attributes of the inference data from the participants by applying participant-specific obfuscation neural networks; the second approach protects the raw forms of the training data contributed by the participants by applying multiplicative random projection; the third approach protects the differential privacy of the contributed training data via additive perturbation. These three approaches are computationally lightweight and can be executed by resource-limited edge devices including smartphones and even mote-class sensor nodes. Extensive performance evaluation performed on multiple datasets and real implementations on IoT hardware platforms show the effectiveness and efficiency of these approaches in protecting data privacy while maintaining the learning and inference performance.

Acknowledgments

It has been my utmost privilege to be given this opportunity to pursue a Ph.D. degree. First, I would like to express my sincere gratitude to my supervisor, Prof. Rui Tan for the continuous support throughout my Ph.D. candidature. The countless discussions that I have had with him have played a considerable role in orientating my research direction in the area of privacy-preserving machine learning in IoT. As a fair and honest critic of my work, Prof. Tan has enabled me to have a better understanding of my research area through his insights and encouragement. His expectation of high standard has also been instrumental in raising the quality of my work. I could not have imagined having a better supervisor and mentor during my Ph.D. candidature period.

Next, I would like to thank my collaborators, Prof. Mo Li, Prof. Arvind Easwaran and Prof. Guosheng Lin, for their supports to my Ph.D. project. With long experience and deep understanding on the research, Prof. Li's comments and suggestions have been crucial in improving the practical relevance of my work. Prof. Arvind Easwaran provides strong guidance and helps me develop my ideas. I am also grateful for Prof. Lin's assistance in enabling me to learn the background of deep learning at the start of my candidature. Their encouragement and insightful comments help me to solve hard problems.

Moreover, I thank my thesis advisory committee members, Prof. Yi Li from the School of Physical and Mathematical Sciences and Prof. Dusit Niyato. Their valuable critique and continuous guidance made significant differences on my research works. I appreciate their kind supervisions for the past four years.

Furthermore, I thank my fellow labmates in NTU-IoT group: Chaojie Gu, Zhenyu Yan, Qun Song, Dongfang Guo and Wenjie Luo, for the daily discussions, for the advice of the system implementations, for the sleepless nights we stayed together to catch up

the conference deadlines and for all the fun we have together during the past four years. The life in the past four years is not boring with their sincere accompany.

Last, but not least, I would like to thank my parents and Yan Dai for being my pillar of support throughout my Ph.D. candidature. Their love, care, and encouragement have been the key to sustaining my mental strength throughout my intellectual pursuit these past four years.

Contents

Abstract	v
Acknowledgments	vi
List of Publications	xii
List of Figures	xiv
List of Tables	xvii
List of Abbreviations	xviii
1 Introduction	2
1.1 Background	2
1.2 System and Privacy Models	4
1.3 Existing Solutions	5
1.4 Main Contributions	8
1.4.1 Chapter 3	8
1.4.2 Chapter 4	9
1.4.3 Chapter 5	10
1.5 Comparison of Proposed Techniques	10
1.6 Thesis Organization	11
2 Existing Privacy-Preserving Machine Learning Approaches	12
2.1 Existing Approaches & Limitations	12
2.1.1 Privacy-Preserving Training	15
2.1.1.1 Parameter Transmission-Based Approaches	15
Distributed Machine Learning	16
Model Personalization	17
2.1.1.2 Data Transmission-Based Approaches	17

	Anonymization	17
	Cryptographic Methods	18
	Data Obfuscation	18
	Data Synthesis	20
2.1.2	Privacy-Preserving Inference	21
2.1.2.1	CryptoNets	21
2.1.2.2	Multi-Party Computation (MPC)	21
2.1.2.3	Autoencoder	22
2.1.3	Remark	22
2.2	Chapter Summary	22

3 PriMask: Cascadable and Collusion-Resilient Data Masking for Mobile

	Cloud Inference	24
3.1	Background and Introduction	24
3.2	Problem Statement	28
3.2.1	System Overview and Threat Model	28
3.2.2	Privacy Notation and Attacks	30
3.3	PriMask Design & Implementation	32
3.3.1	Preliminary on HyperNet	32
3.3.2	Split Adversarial Learning Framework	33
3.3.3	Split Adversarial Learning Protocol	34
3.3.4	Generalizability and Implementations	36
3.3.5	A Simple Case Study on MNIST	37
3.4	Human Activity Recognition	40
3.4.1	HAR Dataset, InferNet, and HyperNet	40
3.4.2	Evaluation Results for HAR	41
3.4.2.1	Impact of PriMask on InferNet accuracy	42
3.4.2.2	Resilience against a single colluding mobile	43
3.4.2.3	Resilience against multiple colluding mobiles	46
3.5	Urban Environment Crowdsensing	46
3.5.1	Dataset, InferNet, ExtNet, and PriMask	47

3.5.2	Evaluation Results for UEC	48
3.6	Driver Behavior Recognition	49
3.6.1	DBR Dataset and System Design	50
3.6.2	Evaluation Results for DBR	51
3.7	Discussions	52
3.8	Summary	54
4	On Lightweight Privacy-Preserving Collaborative Learning for Internet of Things by Independent Random Projections	55
4.1	Background and Introduction	55
4.2	Preliminaries	59
4.2.1	Supervised Collaborative Learning	59
4.2.2	Random Gaussian Projection (GRP) and Other Random Projections	60
4.3	Problem Statement and Approach	61
4.3.1	Problem Statement	61
4.3.2	Gaussian Random Projection Approach	65
4.3.2.1	Gaussian random projection	65
4.3.2.2	Deep learning on projected data	66
4.3.3	Illustrating Examples	68
4.3.3.1	A 2-dimensional example	68
4.3.3.2	Impact of inter-class overlaps on learning performance .	70
4.3.3.3	A 10-dimensional example	72
4.3.3.4	Condition numbers of various projection matrices	73
4.3.4	Alternative Approaches and Limitations	74
4.3.4.1	Non-collaborative learning	74
4.3.4.2	Differential privacy	74
4.4	Performance Evaluation	77
4.4.1	Evaluation Methodology and Datasets	77
4.4.2	Evaluation Results with MNIST Dataset	81
4.4.2.1	Impact of N on learning performance	82
4.4.2.2	Classification accuracy of different classes	84

4.4.2.3	Impact of the horizontal distribution of data	84
4.4.2.4	Impact of data compression	85
4.4.2.5	Various random projection approaches	85
4.4.2.6	Impact of DP noises	87
4.4.3	Evaluation Results with Spambase Dataset	89
4.4.4	Evaluation Results with FSD Dataset	91
4.4.5	Evaluation Results with CIFAR-10 Dataset	91
4.4.6	Summary and Discussion	95
4.5	Implementation and Benchmark	96
4.6	Chapter Summary	98
5	Differentially Private Collaborative Learning for the IoT Edge	99
5.1	Background and Introduction	99
5.2	Approach Design	101
5.2.1	System Model	101
5.2.2	Approach Overview	102
5.2.3	Achieving ϵ -Differential Privacy	104
5.3	Performance Evaluation	106
5.3.1	Evaluation Methodology and Settings	106
5.3.2	Evaluation Results	107
5.4	Chapter Summary	109
6	Conclusion and Future Research	111
6.1	Conclusion	111
6.2	Future Research	112
6.2.1	Privacy-Preserving Unsupervised Learning	112
6.2.2	Adversarial Protection Schemes	113
	References	115

List of Publications

Book chapter

[1] **Linshan Jiang** and Rui Tan. Lightweight Privacy-Preserving Machine Learning and Inference. Intelligent IoT for the Digital World (Eds. Yang Yang, Xu Chen, Rui Tan, Yong Xiao). Wiley. 2021.

Journal papers

[1] **Linshan Jiang**, Rui Tan, Xin Lou and Guosheng Lin. On Lightweight Privacy-Preserving Collaborative Learning for Internet of Things by Independent Random Projections. ACM Transactions on Internet of Things (TIOT), vol. 2, no. 11, pp 1–32, May 2021.

[2] **Linshan Jiang**, Rui Tan, Arvind Easwaran. Resilience Bounds of Network Clock Synchronization with Fault Correction. ACM Transactions on Sensor Networks (TOSN). Article No. 38, pp. 1-30, Sept. 2020.

[3] Dixing Xu, Mengyao Zheng, **Linshan Jiang**, Chaojie Gu, Rui Tan and Peng Chen. Lightweight and Unobtrusive Data Obfuscation at IoT Edge for Remote Inference. IEEE Internet of Things Journal, vol. 7, no. 10, pp. 9540-9551, Oct. 2020.

[4] Chaojie Gu, **Linshan Jiang**, Rui Tan, Mo Li and Jun Huang. Attack-Aware Synchronization-Free Data Timestamping in LoRaWAN. ACM Transactions on Sensor Networks (TOSN), 32 pages, 2021. Accepted, in press.

[5] Yang Zhao, Jun Zhao, **Linshan Jiang**, Rui Tan, Dusit Niyato, Zengxiang Li, Lingjuan Lyu and Yingbo Liu, Privacy-Preserving Blockchain-Based Federated Learning for IoT Devices. IEEE Internet of Things Journal, vol. 8, no. 3, pp. 1817-1829, Feb. 2021.

Conference and workshop papers

[1] **Linshan Jiang**, Rui Tan, Xin Lou, and Guosheng Lin, On Lightweight Privacy-Preserving Collaborative Learning for Internet-of-Things Objects. The 4th ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI), 12 pages, Montreal, Canada. CPS-IoT Week 2019.

[2] **Linshan Jiang**, Xin Lou, Rui Tan, and Jun Zhao. Differentially Private Collaborative Learning for the IoT Edge. The 2nd International Workshop on Crowd Intelligence for Smart Cities: Technology and Applications (CICS), co-located with the International Conference on Embedded Wireless Systems and Networks (EWSN), 6 pages, Feb 25, 2019, Beijing, China.

[3] Mengyao Zheng, Dixing Xu, **Linshan Jiang**, Chaojie Gu, Rui Tan and Peng Chen. Challenges of privacy-preserving machine learning in IoT. The 1st International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things (AIChallengeIoT), co-located with the 17th ACM Conference on Embedded Networked Sensor Systems (SenSys), 6 pages, November 10, 2019, New York, USA.

[4] Rui Tan, **Linshan Jiang**, Arvind Easwaran and Jothi Prasanna Shanmuga Sundaram. Resilience bounds of sensing-based network clock synchronization. The 24th IEEE International Conference on Parallel and Distributed Systems (ICPADS), 9 pages, December 11-13, 2018, Sentosa, Singapore.

[5] Chaojie Gu, **Linshan Jiang**, Rui Tan, Mo Li, Jun Huang. Attack-Aware Data Timestamping in Low-Power Synchronization-Free LoRaWAN. The 40th IEEE International Conference on Distributed Computing Systems (ICDCS), 12 pages, July 8-10, 2020, Singapore.

[4] Chaojie Gu, **Linshan Jiang** and Rui Tan. LoRa-Based Localization: Opportunities and Challenges. The 1st Workshop on Low Power Wide Area Networks for Internet of Things (LPNET), co-located with the International Conference on Embedded Wireless Systems and Networks (EWSN), 6 pages, Feb 25, 2019, Beijing, China.

Paper under review

[1] **Linshan Jiang**, Qun Song, Rui Tan and Mo Li. PriMask: Cascadable and Collusion-Resilient Data Masking for Mobile Cloud Inference. submitted to a conference for peer review.

List of Figures

1.1	Three-layer computing stack model in IoT.	3
1.2	System model.	4
2.1	The hierachical taxonomy of privacy-preserving machine learning approaches.	13
3.1	Properties of heterogeneous MaskNets.	27
3.2	System model.	29
3.3	Inversion attack on MNIST dataset.	31
3.4	Split adversarial learning (SAL) framework for training the HyperNet used to generate MaskNets. In SAL, the PSP needs unlabeled training data \mathbf{x} , which can be from an open dataset or the ISP as illustrated in the figure.	32
3.5	HyperNet architecture.	38
3.6	Impact of PriMask on MNIST test accuracy and privacy protection. λ is adversarial learning factor ($\lambda = 0$ means that adversarial learning is not enabled).	38
3.7	Original, masked, reconstructed samples. (c) and (e) show ISP's reconstructions with the smallest MSEs when adversarial learning factor λ is 0 and 0.6.	39
3.8	Impact of PriMask on HAR test accuracy and privacy protection. Legends denoted by 'r' and 'p' are for HyperNets adversarially trained with inversion attack and private attribute extraction, respectively.	42
3.9	Traces of raw and ISP's reconstructed data for the first axis of linear acceleration, as well as the first dimension of masked data for a certain non-colluding mobile. The ground truth human activity is standing. . . .	44

3.10	Impact of ISP's ExtNet architecture and number of colluding mobiles on ASR. In (a), ExtNet#1 is also the architecture used by PSP in SAL. Whiskers of an error bar represent maximum and minimum.	45
3.11	Impact of PriMask on UEC test accuracy and privacy protection. Legends denoted by 'r' and 'p' are for HyperNets adversarially trained with inversion attack and privacy extraction.	49
3.12	Impact of PriMask on DBR accuracy & privacy.	51
3.13	Original, masked, reconstructed samples. The inversion MSE for (d) is the smallest among all non-colluding mobiles, i.e., this example is the worst case for non-colluding mobiles. Only <i>a priori</i> global information of the dataset (e.g., rough contours of car window and driver) can be seen from reconstructed samples, which are not specific to a certain driver and thus not private. In fact, ISP can generate an image similar to (d) by averaging all training samples.	52
4.1	A collaborative learning system.	61
4.2	Two-dimensional example. Original data vectors and projected data vectors (red: class 0; blue: class 1). The ranges for the x and y axes are $[-10, 10]$	69
4.3	Test accuracy based on projected data vs. the number of participants. . .	70
4.4	Impact of inter-class overlaps.	71
4.5	Test accuracy based on projected data vs. the condition number.	72
4.6	The distributions of the condition number of Gaussian, Rademacher, binary random matrices with dimension 28×28	73
4.7	Example images from MNIST dataset.	79
4.8	CNN with a projected MNIST image as input.	82
4.9	Impact of the number of participants (MNIST). The error bars represent min and max.	82
4.10	The F1 scores of different handwritten digits in the MNIST dataset under the CNN and the GRP-DNN approaches (MNIST).	83

4.11	Four horizontal distributions of the training data among 10 participants. The test accuracies for the four distributions are 96.17%, 96.33%, 96.24%, 96.32%, respectively (MNIST).	84
4.12	Impact of data compression on learning performance (MNIST, $N = 100$).	85
4.13	The test accuracy of GRP-DNN, BRP-DNN, RRP-DNN when N varies (MNIST).	86
4.14	The test accuracy of GRP-DNN, BRP-DNN, RRP-DNN when the com- pression ratio varies (MNIST, $N = 100$).	86
4.15	Impact of privacy loss of DP and LDP on learning performance (MNIST).	87
4.16	Impact of the number of participants (spambase). The error bars represent min and max.	89
4.17	CIFAR-10 image samples. The classes are airplanes, cars, birds, cats, deers, dogs, frogs, horses, ships, and trucks.	90
4.18	Structure of CNN for FSD recognition.	92
4.19	Impact of the number of participants on the learning performance (FSD). The error bars represent min and max.	92
4.20	Impact of the number of participants on the learning performance (CIFAR- 10).	93
4.21	Impact of data compression (CIFAR-10, $N = 1$).	94
4.22	Impact of differential privacy loss (CIFAR-10).	94
5.1	Overview of the proposed privacy-preserving collaborative learning approach.	103
5.2	CNN structure.	107
5.3	Impact of privacy loss level ϵ on the test accuracy of the collaboratively learned model with DP.	107
5.4	Impact of batch size on the test accuracy of the collaboratively learned model with DP ($\epsilon = 2$).	109
5.5	Impact of privacy loss level ϵ on the test accuracy of the collaborative learning approach that perturbs the original data for DP.	110

List of Tables

3.1	PriMask applications and benchmark results including model sizes and compute times (unit: ms).	36
3.2	Statistics of InferNet’s test accuracies across 100,000 MaskNets ($\lambda = 0.1$; AttackNet = ExtNet).	42
4.1	The overhead of various approaches.	97

List of Abbreviations

IoT	Internet of Things
ML	Machine Learning
PPCL	Privacy-Preserving Collaborative Learning
SGD	Stochastic Gradient Descent
DML	Distributed Machine Learning
MPC	Multi-Party Computation
HE	Homomorphic Encryption
DP	Differential Privacy
GAP	Generative Adversarial Privacy
GAN	Generative Adversarial Network
ISP	Inference Service Provider
MLE	Maximum Likelihood Estimation
ObfNet	Obfuscation Neural Network
PSP	Privacy Service Provider
SAL	Split Adversarial Learning
AttackNet	Attack Neural Network
MLP	Multilayer Perception
CDF	Cumulative Distribution Function
HAR	Human Activity Recognition
IMU	Inertial Measurement Unit
ReLU	Rectified Linear Unit
ASR	Attack Success Rate
PEM	Pupil Environment Monitoring
DBR	Driver Behavior Recognition
GRP	Gaussian Random Projection
DNN	Deep Neural Network
SVM	Support Vector Machine
RBF	Radial Basis Function
LDP	Local Differential Privacy
RRP	Rademacher Random Projection
BRP	Binary Random Projection
NCL	Non-collaborative Learning
FSD	Free Spoken Digit

MFCC	Mel-frequency cepstral coefficients
BN	Batch Normalization

Chapter 1

Introduction

1.1 Background

With the advances of sensing and communication technologies, the Internet of Things (IoT) will become a main data generation infrastructure in the future. The drastically increasing amount of data generated by IoT will create unprecedented opportunities for various novel applications powered by machine learning (ML). However, various system challenges need to be addressed to implement the envisaged intelligent IoT.

IoT in nature is a distributed system consisting of heterogeneous nodes with distinct sensing, computation, and communication capabilities. Specifically, it consists of massive sensors with small form factors deeply embedded in the physical world, personal devices that move with people, intermediate network such as wireless access points, as well as the cloud backend.

IoT adopts the three-layer computing stack model as shown in Fig. 1.1. The model consists of *edge computing* layer, *fog computing* layer and *cloud computing* layer. The edge computing layer is based on the IoT end devices and performs computation at the data sources. Most time-sensitive data is analyzed on the edge nodes. For example, a smart home sensor can detect unsafe high temperatures and smoke in a house and then raise fire alarm. In the fog computing layer, the device with computing, storage, and network connectivity can be a fog node. Examples include various controllers, switches, routers, embedded servers, and IoT gateways. The fog computing layer analyzes the IoT data close to where it is collected to minimize latency and avoid the transmissions of massive data to the Internet's computing core. The core part is the cloud computing

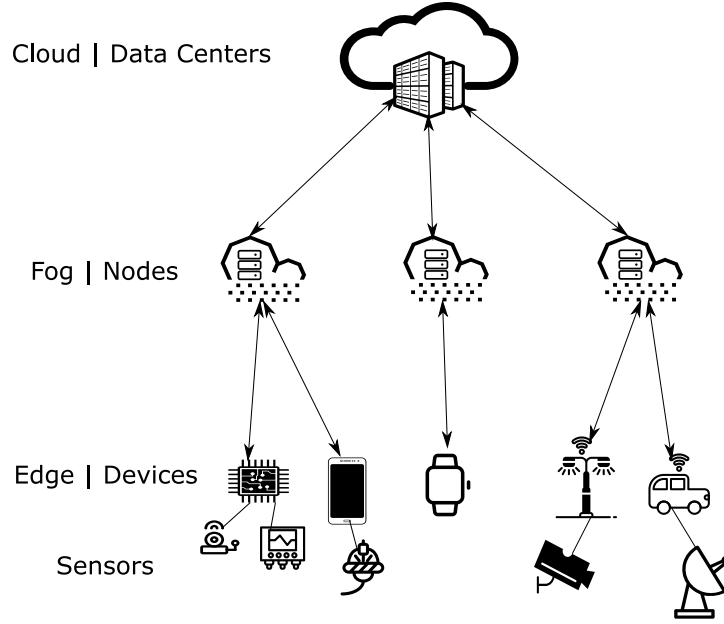


Fig. 1.1: Three-layer computing stack model in IoT.

layer. In this layer, the resourceful backend can deal with the summarized IoT data transmitted from the previous layers. It saves the cost by avoiding purchasing expensive systems and equipment for the local network.

The separation of data sources and the computation power needed for processing massive data is a key characteristic of IoT. Most IoT data are generated by the end devices that often have limited computation resources, while the computation power needed by ML model training and execution are located at the fog nodes and in the cloud. Besides, the communication channels between the end devices and the cloud are often constrained, in that they are limited in bandwidth and of intermittent availability and considerable latencies. Due to this separation, to enable advanced data processing for better intelligence, a possible approach is to transmit the data from the data sources to the higher layers with sufficient computation power for advanced ML processing.

However, the above approach engenders privacy concerns. As the end devices can be deeply embedded in the user's private spaces and times, the data generated by these end devices can contain privacy-sensitive information. To gain wide acceptance, the IoT-ML fabric must respect the users' privacy. The lack of privacy preservation may even go against the recent legislation such as the General Data Protection Regulation in

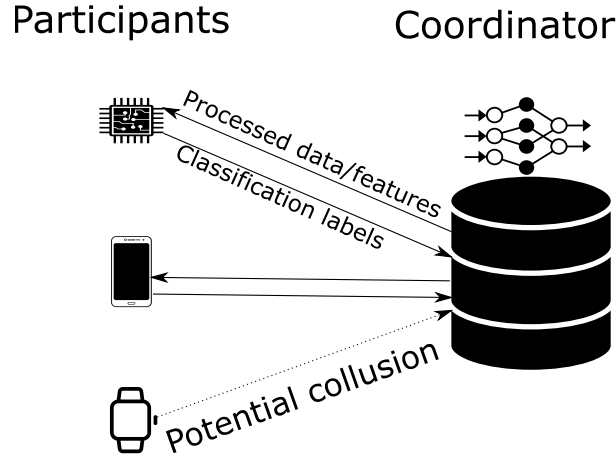


Fig. 1.2: System model.

European Union. However, privacy preservation often presents substantial challenges to the system design.

1.2 System and Privacy Models

To address the privacy concerns, this thesis mainly considers a distributed ML system consisting of multiple participants and a coordinator in the context of IoT. Fig. 1.2 illustrates the distributed ML system. In this thesis, the participants are resource-constrained data generators with many training and/or inference data samples. The coordinator is an honest-but-curious IoT backend with sufficient computation power to orchestrate the ML processes. The participants and coordinator collaborate to realize a classification system in the context of IoT. As the learning process is often compute-intensive, most of the learning computation should be accomplished by the coordinator. This thesis focuses on the problem of distributed machine learning while protecting certain privacy contained in the data samples. This thesis mainly focuses on protecting the confidentiality of raw form of the data, while in Chapter 3 the thesis also considers the private attribute contained in the inference data.

The privacy concern regarding the data samples is primarily due to that the data samples may contain information beyond the classification objective in question. For example, consider a privacy-preserving collaborative learning (PPCL) system for training a classifier to recognize human body activity (e.g., sitting, walking, climbing stairs, etc).

The recognition is based on various body signals (e.g., motion, heart rate, breath rate, etc) that are captured by wearable sensors. However, the raw body signals can also be used to infer health statuses of the participants and even pinpoint the patients of certain diseases. This thesis adopts the following threat and privacy models.

- **honest-but-curious coordinator:** This thesis assumes that the coordinator will honestly coordinate the distributed learning process and inference process, aiming to realize the best ML performance. Thus, it will neither tamper with any data or parameters collected from or transmitted to the participants. However, the coordinator is curious about the participants' privacy contained in the data samples. The coordinator may benefit from the private information irrelevant to the objective of the supervised classifier.
- **Potential collusion between participants and coordinator:** This thesis assumes that the participants are not trustworthy in that they may collude with the coordinator in finding out other participants' privacy contained in the data samples. The colluding participants are also honest, i.e., they will faithfully contribute their training data to improve the supervised classifier. The design of the distributed system should maintain the privacy preservation for a participant when any or all other participants are colluding with the coordinator. In this thesis, the potential collusion among participants is not considered, since the privacy concern of the participants is mainly from the honest-but-curious cloud.
- **Label privacy:** The class labels may also contain information about the participant. This thesis does not consider label privacy. This thesis assumes that the participant willingly contributes the labeled data samples and should have no expectation of privacy regarding labels. In the future research, I will consider how to protect the label privacy of the participants with unsupervised learning, which will be discussed in Chapter 6.

1.3 Existing Solutions

Privacy-preserving ML has received extensive research in the context of cloud computing. Thus, it is of great interest to investigate whether the existing solutions can

be applied in the context of IoT. To this end, this section provides a brief review of the existing privacy-preserving ML approaches, which are classified into two categories: *privacy-preserving training* and *privacy-preserving inference*. The nodes in a privacy-preserving ML system often have two roles: *participant* and *coordinator*. The participants are often the data generators (e.g., smartphones), whereas the coordinator (e.g., a cloud server) orchestrates the ML process. The *privacy-preserving training* schemes aim to learn a global ML model or multiple local ML models from disjoint local datasets which, if aggregated, would provide more useful/precise knowledge. Thus, the primary objective of privacy protection is to preserve the privacy of the data used for building an ML model in the training phase. Differently, *privacy-preserving inference* schemes focus on the scenario where a global ML model at the coordinator has been trained and the participants transmit the unlabeled data to the coordinator for inference. The aim is to protect the privacy of the input data in the inference phase and maintain the inference accuracy. Since the computation and communication overheads are the key considerations in the design of IoT systems, ML for IoT should be of low overhead and with privacy preservation.

In a privacy-preserving training process orchestrated by the coordinator, the participants collaboratively train a global model from their disjoint training datasets while the privacy of the training datasets is preserved. Distributed machine learning (DML) [1–6] is a typical scheme of this category, in which only the model parameters are exchanged among the nodes. However, the local model training and the iterative information exchanges are compute- and communication-intensive. Recently, the federated learning scheme [7] has received wide research interests. Federated learning is a typical type of DML. During the training process of federated learning, the training data possessed by each participant is not exchanged. Instead, only the gradients of the model trained by each participant are transmitted to the coordinator. Thus, federated learning is considered promising for privacy preservation. However, it requires each participant to train a local model based on the training data it possesses. The intensive computation of the local training renders federated learning ill-suited for IoT devices especially for the battery-powered devices to act as participants. Therefore, federated learning is mostly studied under the context where each participant is an enterprise that has sufficient computation power. Besides, there exist a few examples of federated learning on the mobile

devices, such as Google Fed-GBoard [8]. However, Google Fed-GBoard requires at least 2 gigabytes of memory available and it only works when the mobile device is charging, connected to an un-metered network, and idle without other tasks.

If the training data samples are to be transmitted to the coordinator, they can be obfuscated or encrypted for data privacy protection. Obfuscation is often achieved via additive perturbation and multiplicative projection. Additive perturbation implemented via Laplacian [9], exponential [10], and median [11] mechanisms can provide differential privacy [12]. Multiplicative projection [13,14] protects the confidentiality of the raw forms of the original data. In [13], the participants use distinct secret projection matrices, where the Euclidean distances among the projected data samples are no longer preserved. This can degrade the performance of distance-based ML algorithms. To address this issue, in [13], the participants need to project a number of public data samples and return the results to the coordinator that will learn a regress function to preserve Euclidean distances. However, this approach is only applicable to distance-based classifier. The conventional classifier does not scale well with the volume of the training data and the complexity of the data patterns. ML can be also performed based on homomorphically encrypted data samples [15–18]. However, homomorphic encryption incurs high compute overhead compared with additive perturbation and multiplicative projection.

Privacy-preserving inference approaches assume that the ML model at the coordinator has been previously trained using public plaintext data. They aim to protect the privacy contained in test data samples while maintaining the inference accuracy. Additive perturbation is generally not advisable for deep models because the inference accuracy of deep models can be significantly degraded by small perturbations on input data [19]. In order to achieve privacy preservation in the inference phase against an honest-but-curious coordinator running the ML model, CryptoNets [20] and Multi-party Computation (MPC) [21] are proposed. CryptoNets [20] adjusts the feed-forward neural network trained with plaintext data so that it can be applied to the homomorphically encrypted data to make encrypted inference. Unfortunately, the high computational complexity of homomorphically encrypted renders CryptoNets unpractical for IoT devices. Moreover, although CryptoNets does not need to support training over ciphertext, the neural network still needs to satisfy certain conditions. MPC [22] enables the parties involved to jointly compute a function over their inputs while keeping those inputs private.

The work [21] applies MPC to achieve *privacy-preserving inference*. However, MPC requires many rounds of communication between the participant and the coordinator, representing considerable communication overhead.

1.4 Main Contributions

To address the computation and communication overheads with privacy preservation approaches, this thesis proposes and studies three novel privacy preservation techniques for deep learning and inference. The first scheme is a privacy-preserving inference approach. It is a scalable, *cascadable*, and collusion-resilient privacy preservation approach for mobile devices to use the cloud inference services. By executing a small-scale neural network on the mobile devices to obfuscate inference data, the privacy contained in the inference data is preserved. The second scheme is a lightweight privacy-preserving collaborative learning (PPCL) scheme. It applies independent Gaussian random projection at each IoT object to obfuscate data and trains a deep neural network at the coordinator based on the projected data from the IoT objects. This approach introduces light computation overhead to the IoT objects and moves most workload to the coordinator that can have sufficient computing resources. The last scheme is also a PPCL approach, in which the fog nodes and the cloud train different stages of a deep neural network, and the data transmitted from a fog node to the cloud is perturbed by Laplacian random noises to achieve ϵ -differential privacy. The three proposed approaches are all designed for an IoT system consisting of distributed IoT nodes with limited computation powers.

The main contributions of each chapter are summarized as follows.

1.4.1 Chapter 3

Mobile cloud offloading is indispensable for inference tasks based on large-scale deep models. However, transmitting privacy-rich inference data to the cloud incurs concerns. This chapter presents the design of a system called *PriMask*, in which the mobile device uses a secret small-scale neural network called *MaskNet* to mask the data before transmission. *PriMask* significantly weakens the cloud's capability to recover the data or extract certain private attributes. The *MaskNet* is *cascadable* in that the mobile can opt in to

or out of its use seamlessly without any modifications to the cloud’s inference service. Moreover, the mobiles use different MaskNets, such that the collusion between the cloud and some mobiles does not weaken the protection for other mobiles. We devise a *split adversarial learning* method to train a neural network that generates a new MaskNet quickly (within two seconds) at run time. We apply PriMask to three mobile sensing applications with diverse modalities and complexities, i.e., human activity recognition, urban environment crowdsensing, driver behavior recognition. Results show PriMask’s effectiveness in all three applications.

1.4.2 Chapter 4

The Chapter 4 of this thesis considers the design and implementation of a practical privacy-preserving collaborative learning scheme, in which a curious learning coordinator trains a better machine learning model based on the data samples contributed by a number of IoT objects, while the confidentiality of the raw forms of the training data is protected against the coordinator. Existing distributed machine learning and data encryption approaches incur significant computation and communication overhead, rendering them ill-suited for resource-constrained IoT objects. This chapter studies an approach that applies independent random projection at each IoT object to obfuscate data and trains a deep neural network at the coordinator based on the projected data from the IoT objects. This approach introduces light computation overhead to the IoT objects and moves most workload to the coordinator that can have sufficient computing resources. Although the independent projections performed by the IoT objects address the potential collusion between the curious coordinator and some compromised IoT objects, they significantly increase the complexity of the projected data. This chapter leverages the superior learning capability of deep learning in capturing sophisticated patterns to maintain good learning performance. Extensive comparative evaluation shows that this approach outperforms other lightweight approaches that apply additive noisification for differential privacy and/or support vector machines for learning in the applications with light to moderate data pattern complexities.

1.4.3 Chapter 5

The Chapter 5 of this thesis presents the design of a privacy-preserving collaborative learning approach, in which the edge devices and the cloud train different stages of a deep neural network, and the data transmitted from an edge device to the honest-but-curious cloud is perturbed by Laplacian random noises to achieve ϵ -differential privacy. The proposed approach is evaluated on a case study of collaboratively training a convolutional neural network for handwritten digit recognition. The results show that the proposed approach maintains 99% and 96% classification accuracy in implementing privacy loss levels of $\epsilon = 5$ and $\epsilon = 2$, respectively.

1.5 Comparison of Proposed Techniques

Basic requirement of privacy protection is to protect the confidentiality of raw forms of the data samples. Besides that, Chapter 3 considers a cloud inference system in which the mobile sends the inference sample to the Inference Service Provider (ISP) and receives the inference result. In this case, since the mobiles are often used in private spaces and times, the inference samples may contain privacy-sensitive information. This thesis proposes a PriMask approach to obfuscate the inference data before transmission. The approach does not require any changes to the cloud inference system that is designed to handle plaintext inference data samples. Differently, Chapter 4 and Chapter 5 consider privacy-preserving collaborative learning, which aims to protect the privacy contained in the training data samples. Chapters 4 and 5 assume that the training data are sensed and collected by IoT end devices in private spaces and times, whereas Chapter 3 assumes that the ISP has pretrained a deep model with public or self-collected dataset. Thus, the privacy of the training data is not considered. In summary, the proposed approach in Chapter 3 is more suitable for the existing mobile inference applications while the proposed approaches in Chapters 4 and 5 are to build new inference systems from scratch.

The main difference of the approaches in Chapters 4 and 5 is on the application scenario. Chapter 4 considers the training data sensed by the mote-class sensors and resource-constraint IoT end devices. Therefore, Chapter 4 adopts the random projection

scheme to project the training data since the random projection operation on these devices is feasible. Differently, Chapter 5 considers the training data contributed by the more resourceful edge devices such as smartphones. In these edge devices, some layers of a neural network model can be trained. To utilize the computation power of such edge devices, Chapter 5 proposes a split training approach with data noisification to preserve the privacy in the transmitted features.

1.6 Thesis Organization

The remaining part of this thesis is organized as follows. Chapter 2 reviews state-of-the-art privacy-preserving machine learning approaches in the context of cloud computing and present the taxonomy of these approach in the context of IoT. Chapter 3 presents a lightweight privacy-preserving inference approach with PriMask to obfuscate the inference data. Chapter 4 presents a lightweight privacy-preserving training approach based on the random projection to obfuscate the training data. Chapter 5 presents a lightweight privacy-preserving training approach to obfuscate transmitted features, which fulfills the requirement of the differential privacy. Finally, the thesis is concluded in Chapter 6, with discussions on relevant future directions.

Chapter 2

Existing Privacy-Preserving Machine Learning Approaches

2.1 Existing Approaches & Limitations

Fig. 2.1 illustrates the taxonomy of the existing privacy-preserving ML schemes. The nodes in a privacy-preserving ML system often have two roles: *participant* and *coordinator*. The participants are often the data generators (e.g., smartphones), whereas the coordinator (e.g., a cloud server) orchestrates the ML process. Since ML has two phases, i.e., training and inference, this chapter classifies the existing approaches into two groups at the top level. The *privacy-preserving training* schemes (§2.1.1) aim to learn a global ML model or multiple local ML models from disjoint local datasets which, if aggregated, would provide more useful/precise knowledge. Thus, the primary objective of privacy protection is to preserve the privacy of the data used for building an ML model in the training phase. Differently, *privacy-preserving inference* schemes (§2.1.2) focus on the scenario where a global ML model at the coordinator has been trained and the participants transmit the unlabeled data to the coordinator for inference. The aim is to protect the privacy of the input data in the inference phase and maintain the inference accuracy.

The work in this chapter has been published as Mengyao Zheng, Dixing Xu, Linshan Jiang, Chaojie Gu, Rui Tan and Peng Chen. Challenges of privacy-preserving machine learning in IoT. The First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things (AIChallengeIoT), New York, NY, 2019, co-located with ACM SenSys.

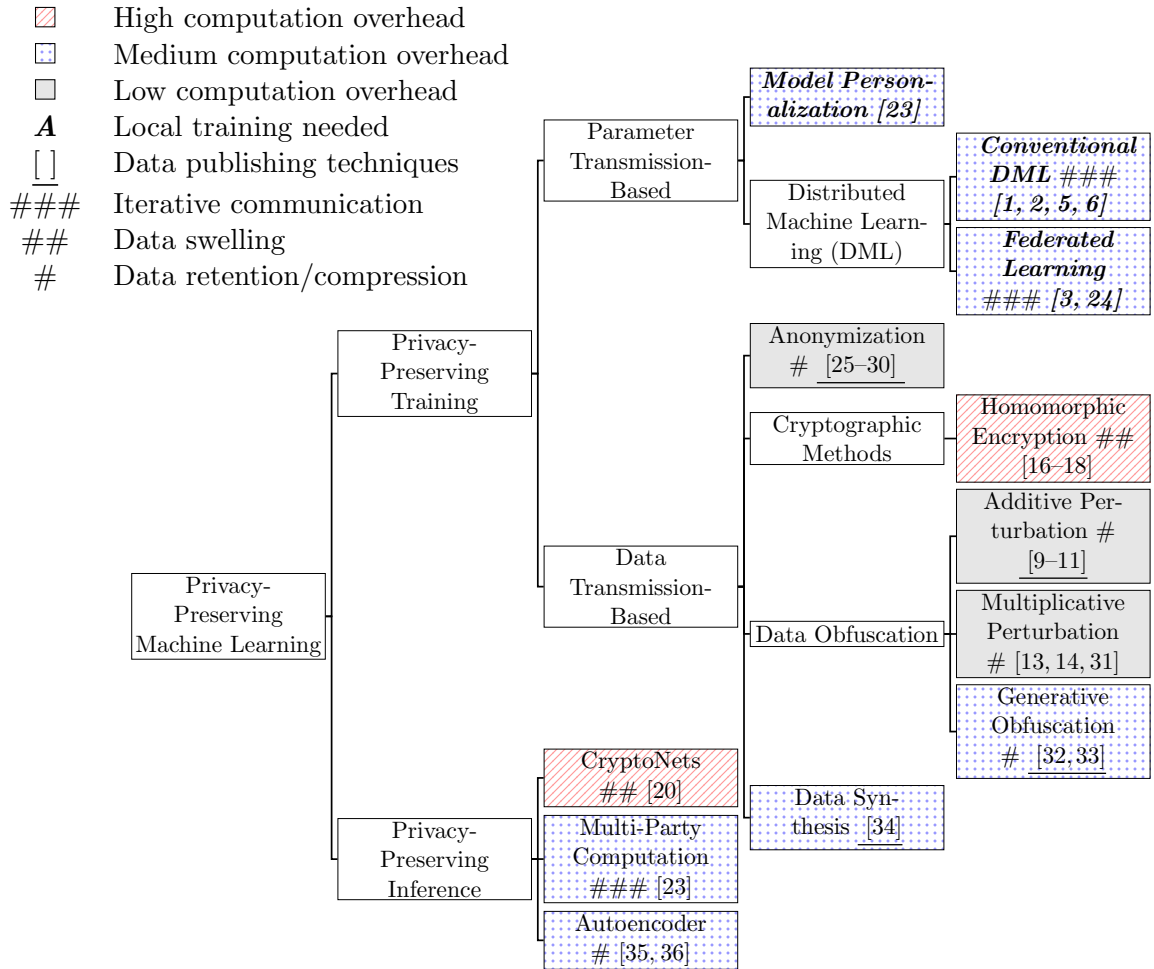


Fig. 2.1: The hierachical taxonomy of privacy-preserving machine learning approaches.

Privacy-preserving training schemes can be further classified based on whether the privacy-sensitive training data samples are transmitted or only the model parameters are transmitted for model training. Parameter transmission-based approaches (§2.1.1.1) include distributed machine learning and model personalization. The approaches that need to transmit local data samples (§2.1.1.2) can be classified into *anonymization*, *cryptographic methods*, *obfuscation*, and *data synthesis* according to the processing made on the training data. (1) Anonymization approaches de-identify data records but do not change the data of interest for model training. (2) Cryptographic methods apply cryptographic primitives to encrypt the data transmitted. (3) Obfuscation methods transform the training data vectors through additive perturbation, multiplicative perturbation, and generative obfuscation. (4) Data synthesis generates a new dataset that resembles the original dataset. Note that some of the data transmission-based approaches for privacy-preserving training are *data publishing* techniques, which focus on the proper sharing of the data or query results and in general do not explicitly address the problem of ML model training. The taxonomy in this chapter includes them for the completeness of the related work review.

Most existing privacy-preserving ML approaches were designed in the context of cloud computing. The participants are often resource-rich nodes from smartphones to cloud servers. In particular, the overhead of communications is not a key concern due to the availability of high-speed connections (e.g., wireline networks and 4G cellular networks). Differently, in the context of IoT, the participants are often resource-constrained devices. Moreover, the communication links among them are generally constrained. Therefore, in the review of the privacy-preserving training (§2.1.1) and inference (§2.1.2) approaches, their computation overhead and communication overhead will be qualitatively discussed. Here are the qualitative labels used in this chapter regarding the computation and communication overheads:

- **Computation overhead:** This chapter classifies the level of computation overhead into *high*, *medium*, and *low*, with homomorphic encryption, neural network training/inference, and additive/multiplicative noisification as the representative examples, respectively. The high-overhead computation tasks are in general infeasible for IoT end devices. For the medium-overhead computation, IoT end devices are

increasingly capable of neural network inference computation due to the emerging inference chips such as Google’s Edge TPU [37]. However, neural network training is still largely infeasible for IoT end devices at present.

- **Communication overhead:** This chapter classifies the communication overheads of the existing approaches into three categories: *iterative communication*, *data swelling*, *data retention/compression*, with distributed machine learning, homomorphic encryption, and additive/multiplicative perturbation as the representative examples, respectively. Specifically, distributed machine learning requires iterative model parameter exchanges among the training participants. Such iterative communications will cause significant challenges for IoT networks due to the bandwidth-limited and intermittent communication links. The ciphertexts produced by homomorphic encryption algorithms often have higher data volumes than the plaintexts. In contrast, the additive and multiplicative perturbation will retain and even reduce the data volumes.

Storage overhead is also a factor of concern for IoT end devices. However, this chapter primarily focuses on computation and communication overhead.

2.1.1 Privacy-Preserving Training

The latest privacy-preserving training approaches that leverage distributed privacy-sensitive data to construct a global ML model or multiple local ML models can be divided into parameter transmission-based (§2.1.1.1) and data transmission-based (§2.1.1.2) techniques.

2.1.1.1 Parameter Transmission-Based Approaches

Approaches of this category transmit model parameters instead of data samples for model training. In this way, parameter transmission-based approaches to privacy-preserving training push computation towards participants rather than the coordinator.

Distributed Machine Learning Distributed Machine Learning (DML) is a representative approach of this category. In DML [1–6, 24], data owners do not reveal their own datasets to anyone in the training phase. In each iteration, the participants upload merely the locally computed parameters or gradients to the coordinator to achieve *collaborative learning* [38]. Conventional DML algorithms [1, 2, 5, 6] exploit the fact that the optimization algorithms based on stochastic gradient descent (SGD) can be parallelized, if the data held by different participants are independently and identically distributed (i.i.d.). Variants of SGD such as Selective SGD [2], parallel SGD [5], Alternating Direction Method of Multipliers SGD [6] and Downpour SGD [4] are normally used to update the model weights in the distributed fashion.

Federated learning [3, 24, 39] is another prevailing DML approach with more generalized assumptions. In each iteration, a fraction of participants are randomly selected to retrain the model with their local data using the current global model as the starting point and then individually upload the local stochastic gradient descents. The coordinator will then average the gradient descents and update the global model. However, while federated learning [3, 24, 39] manages to reduce communication overhead, it increases the local computation overhead.

Arguably, model parameters contain some information about the local training data. Therefore, in [1, 2, 24], differential privacy [12] is achieved by adding noises to the locally computed parameter updates. Such schemes thwart definitive inferences about an individual participant if an adversary intentionally collects the obfuscated model updates. Besides, secure data aggregation is applied to aggregate the updates from individual participants [40]. Phong et al. [41] also use additively homomorphic encryption [15] to encrypt model parameters in the federated learning scheme to prevent information leakage. Secure multi-party computation (MPC) is also applied in the federated learning scheme [42–44]. In the subsection 2.1.2.2, the limitation of MPC-based approach is discussed.

Limitations: First, training a deep model locally may be impractical for resource-constrained IoT devices. The paper [39] proposes a secured and privacy-preserving smart home architecture implementing federated learning in which the dedicated computers in home are the federated learning trainers. Second, many iterations are required for the

learning process to converge in these approaches, which results in substantial communication overhead. Due to the computation and communication overhead incurred, DML is mainly deployed in the context of cloud computing with enterprise settings. As shown in [45, 46], federated learning is vulnerable to backdoor attacks. Hitaj et al. [47] devise a powerful attack based on generative adversarial networks [48] for local data recovery. The attack is still effective even when the communicated parameters are perturbed for differential privacy and secure multi-party computation [22] is applied.

Model Personalization The aim of model personalization [23] is not to learn a global model from privacy-sensitive data owned by the participants, but to learn a personal model for each participant based on a public model trained with public data as the starting point. Specifically, the public model is firstly trained with a set of public data at the coordinator and then distributed to each participant. Then, each participant retrains the model with local data. The idea of transfer learning [49] is leveraged to achieve better performance than the model training with merely local data.

Limitations: This approach may be ineffective for some tasks where the local classes have significantly different patterns from the public data. Additionally, the local training is unsuitable for resource-constrained IoT devices.

2.1.1.2 Data Transmission-Based Approaches

This category of approaches allows participants to send local data samples to the honest-but-curious coordinator, while protects certain aspect or attribute of the data samples, e.g., user identity, data contents, or raw form of data. It has the following sub-categories: anonymization, cryptographic methods, data obfuscation, and data synthesis.

Anonymization Anonymization techniques are designed to anonymize the participant's identity in a group of users, changing the value of quasi-identifiers and removing explicit identifiers. Since the aim is to remove the association between data entries and the data owner, the data samples of interest used for model training remains unchanged. For field-structured data, anonymization techniques include k -anonymity [25], l -diversity [26], and t -closeness [27]. The k -same family of algorithms [28–30] are designed to de-identify face images.

Limitations: As analyzed in [26], anonymization techniques are vulnerable to *homogeneity attack* and *background knowledge attack*. Furthermore, anonymization techniques are traditional data publishing techniques proposed for a centralized database. As commented in a survey [50], these anonymization techniques in crowdsensing applications have a main drawback of the need for a trusted proxy to produce the anonymized values and send them to each participant. This implies the risk of single point of failure.

Cryptographic Methods Cryptographic methods encrypt the training data before transmission. However, traditional cryptographic methods suffer from high computation complexity and the sophistication of key management [51]. The method of Homomorphic Encryption (HE) [15] does not need key propagation and has attracted research interest. With HE, computation on ciphertexts generates an encrypted result which matches the result of the operations performed on the plaintext data after decryption. In [16–18], the ML model is trained at the coordinator on the HE ciphertexts. During the inference phase, the data is also encrypted before transmission.

Limitations: However, in HE, operations on the ciphertexts are required to be expressed as polynomials of a bounded degree. Thus, HE is normally applied to the operations with a linear discrimination nature. Furthermore, HE involves intensive computation and leads to *data swelling*. HE causes computation overhead millions times higher than a multiplicative obfuscation approach that will be discussed shortly in Chapter 4. Additionally, HE will make the training process at least an order of magnitude slower [20].

Data Obfuscation Data obfuscation methods perturb the data samples used for training a global model. These methods include additive perturbation, multiplicative perturbation, and generative obfuscation.

- **Additive Perturbation:** Normally, additive obfuscation is often associated with Differential Privacy (DP) [12]. DP is a formal and quantifiable measure of privacy protection, which can be incorporated in data mining and data publishing [52]. The key idea of differentially private data mining [53–55] is to learn a model with plaintext data but perturb the value computed in a certain step (e.g., gradients in

optimization) with noises during the training. Such techniques, as discussed earlier, are often used in DML (§2.1.1.1). Differentially private data publishing aims to output aggregate information without revealing any specific entry. It can be achieved by adding noises to the query results using Laplacian [9], exponential [10], and median [11] mechanisms.

Limitations: Differentially private data publishing techniques are often used to support the release of limited data representations, such as contingency tables or histograms [52]. The amount of noise increases dramatically when the queries are correlated [52]. Besides, differentially private data publishing often caters into the setting of centralized systems, where a curator collects all the data and respond to queries. But such a trusted curator can be questionable and costly in the context of IoT. Moreover, it incurs the risk of single-point failure. If the curator is not available, each data contributor perturbs its own result, which leads to an aggregated noise that significantly exceeds the required amount to ensure ϵ -DP of the final result. Besides, in practical scenarios [47, 56], it has been proved that the trade-off between model usability and privacy is not easy to balance such that it limits the implementation of the DP-based approach in a real application.

- **Multiplicative Perturbation:** Random projection [13, 14, 31] is a typical multiplicative perturbation. Some random projection schemes [31] preserve the dimensionality of the data but are susceptible to *approximate reconstruction attack* [57]. Other schemes [13, 14] reduce the dimensions of the data to better preserve privacy. The approach in [14] standardizes the projection matrix R for all participants. However, this design may scale poorly since any collusion between participants would breach data privacy. In [13], participants use different private matrices for random projection. Therefore, the Euclidean distances for the perturbed data are no longer preserved, which can significantly degrade the classification accuracy for distance-based classifiers. To tackle this problem, in [13], the coordinator uses regression to reconstruct the pairwise distances between the original data vectors based on each participant's obfuscated projection results of a set of public data samples. However, the coordinator can use the public samples and their projections to recover random projection matrix of each participant.

Limitations: In summary, there exists a trade-off between privacy and utility when applying multiplicative perturbation. If each participant uses a different random projection matrix, privacy can be better preserved but classification accuracy in the cloud is likely degraded.

- **Generative Obfuscation:** Different from additive/multiplicative perturbation techniques, generative models can also produce obfuscated data. Huang et al. [32] propose a Generative Adversarial Privacy (GAP) algorithm, which is composed of a *privatizer* and an *adversary* network. GAP formulates a minimax game-theoretic problem where the *privatizer* aims to obfuscate the original data X to render a specified privacy-sensitive attribute Y non-classifiable by the *adversary* network. The *privatizer* and *adversary* are trained in an iterative manner. A recent study [33] applies the generative adversarial network (GAN) approach to address the problem of how a data owner publishes labeled training data with certain private attributes preserved while maintaining the utility of the published data for learning.

Limitations: Running a generative model locally for obfuscation incurs high computation overhead. As a data publishing technique, although GAP [32] restricts the l_2 distance between the original data vector and the obfuscated data vector to control the distortion level, there is no guarantee of the utility of the obfuscated data. However, as the GAN training is highly compute-intensive, the approach [33] is suitable for resource-rich data owners.

Data Synthesis Data synthesis methods use generative models that capture the underlying distribution of a private dataset and generate resembling data samples. Such generalization, ideally, would protect individual-specific information. In [34], differentially private k -means clustering is applied on the raw dataset. Then, generative models are trained only on their own cluster using differentially private gradient descent [34].

Limitations: Data synthesis is a data publishing technique and is usually implemented in a centralized database with massive data samples. As generative models often incur high computation overhead, this approach is not suitable for resource-limited IoT devices.

2.1.2 Privacy-Preserving Inference

Compared with a body of research on privacy-preserving training, less work is dedicated to privacy-preserving inference. Privacy-preserving inference approaches assume that the ML model at the coordinator has been previously trained using public plaintext data. They aim to protect the privacy contained in test data vectors while maintaining the inference accuracy. Additive perturbation is generally not advisable for deep models because the inference accuracy of deep models can be significantly degraded by small perturbations on input data [19]. In order to achieve privacy preservation in the inference phase against an honest-but-curious coordinator running the ML model, CryptoNets [20] and Multi-party Computation (MPC) [21] are proposed.

2.1.2.1 CryptoNets

Gilad-Bachrach et al. [20] adjust the feed-forward neural network trained with plaintext data so that it can be applied to the homomorphically encrypted data to make encrypted inference. A secret key is needed to decrypt the result. Through the process, not only the data but also the inference result are kept secret against the honest-but-curious coordinator.

Limitations: Unfortunately, the high computational complexity of HE renders CryptoNets unpractical for IoT devices. Moreover, although CryptoNets does not need to support training over ciphertext, the neural network still needs to satisfy certain conditions. For example, a square polynomial function instead of a sigmoid or ReLU function should be used as the activation function. However, using square polynomial function as the activation function is rare for existing neural networks. Scaling is also required since encryption scheme does not support floating-point numbers.

2.1.2.2 Multi-Party Computation (MPC)

MPC [22] enables the parties involved to jointly compute a function over their inputs while keeping those inputs private. Barni et al. [21] apply MPC in *privacy-preserving inference*. Specifically, the participant encrypts the data and sends it to the coordinator. The coordinator computes an inner product between the data and the weights of the first layer and sends the results back to the participant. Then, the participant applies

decryption and non-linear transformation. Results are again encrypted before being transmitted to the coordinator. The process continues until all the layers have been computed. In this scheme, the input data and the knowledge embedded in the neural networks are both protected.

Limitations: MPC requires many rounds of communication between the participant and the coordinator, representing considerable communication overhead.

2.1.2.3 Autoencoder

Two recent studies [35, 36] apply autoencoder to publish time series data (i.e., accelerator data) for inference. The autencoder is trained such that the encoder can weaken a certain private attribute and the decoder's output that is published largely preserves the raw form of the original data.

Limitations: The autencoder requires moderate computability of excuting an autencoder which is a complete neural network, denoting to considerable computation overhead especially on the edge devices. Besides, their approach needs to retrain the inference model based on the encoded training data, which incurs additional computation on the cloud.

2.1.3 Remark

The existing approaches reviewed in §2.1.1 and §2.1.2 have different threat, privacy and system models. The anonymization and additive/multiplicative perturbation approaches often introduce affordable overheads and thus are feasible in the context of IoT. However, anonymization mainly focuses on private data publishing and does not address the problem of model training. Thus, additive and multiplicative perturbation approaches are promising for privacy-preserving training in IoT. In contrast, lightweight privacy-preserving inference approaches for IoT are lacking.

2.2 Chapter Summary

This chapter reviews the existing privacy-preserving ML approaches that were developed largely in the context of cloud computing and discusses their limitations in the

context of IoT. From the above survey, there is a body of research on privacy-preserving training. Additive and multiplicative perturbation approaches are promising for privacy-preserving training in IoT due to their low computation and communication overhead. In contrast, lightweight privacy-preserving inference received limited research. Chapter 3 of this thesis proposes a lightweight privacy preservation technique for inference.

Chapter 3

PriMask: Cascadable and Collusion-Resilient Data Masking for Mobile Cloud Inference

3.1 Background and Introduction

Recent years have witnessed the forming fabric of machine learning and mobile computing. While running neural networks on resource-constrained mobile devices has received extensive research [58, 59], large-scale neural networks may still incur lengthy execution times that impede user experiences and drain excessive battery energy. For instance, on Huawei P20 Pro, recognizing a face using Inc-ResNet-V1 with hardware acceleration requires 26 seconds [60]. Such neural networks and those beyond the mobiles’ capabilities should be executed in the cloud. Besides technical constraints, neural networks may have high commercial values and design costs (e.g., 1.3 million US\$ cost for training a natural language processing model [61]). Thus, many inference services are proprietary and remain in the owners’ clouds. As such, the *cloud inference* is indispensable.

To use cloud inference, the mobile sends the inference sample to the Inference Service Provider (ISP) and receives the result. As mobiles are often used in private spaces and

The work in this chapter has been submitted as **Linshan Jiang, Qun Song, Rui Tan and Mo Li. PriMask: Cascadable and Collusion-Resilient Data Masking for Mobile Cloud Inference.** to a conference.

times, the samples may contain privacy-sensitive information. For instance, the voice samples for using a virtual assistant contain rich information about the user, e.g., gender, age, mood, and voiceprint. Although the network transmissions can be protected by cryptography against the external eavesdroppers, protecting the user’s privacy against an honest-but-curious ISP while maintaining the accuracy of the cloud inference is a challenging problem. While the ISP honestly executes inference, it may purposely or accidentally extract the users’ privacy.

To achieve privacy-preserving inference, *homomorphic encryption* and *neural network masking* approaches have been proposed. In the homomorphic encryption approaches [16–18, 20], the data owner sends the homomorphically encrypted sample to the ISP for performing inference in the encryption domain. However, for resource-constrained mobiles, homomorphic encryption incurs high computation overhead. For instance, it takes more than ten minutes for a 900 MHz quad-core processor to encrypt a 28×28 grayscale image [62]. Differently, neural network masking views several neural network layers as a data masking operation. Thus, the data owner runs these layers and sends the output to the ISP that runs the inference neural network (InferNet) on the masked data. However, the existing neural network masking approaches [63–67] only counteract the privacy threat from the external eavesdroppers who do not have the details of the masking. In these approaches, since the ISP knows the layers used for masking, it can launch the *model inversion attack* [68, 69] to reconstruct the original data.

Running a small-scale neural network has become feasible on mobiles and even lower-profile wearables. Thus, from the perspective of engineering an workable privacy-respected cloud inference system for mobiles, neural network masking is a promising basis. This chapter designs such a system called *PriMask* with the following four objectives. **First**, different from the existing studies [63–67] that address external eavesdroppers, PriMask considers the privacy threat from the honest-but-curious ISP. **Second**, PriMask is resilient to the collusion between any mobile and ISP, in that the collusion does not weaken the privacy protection for non-colluding mobiles. This collusion-resilient property is important because otherwise the system is susceptible to any compromised individual among many mobiles. **Third**, PriMask does not require any changes to the ISP’s InferNet that was designed without privacy preservation considerations. This frees the ISP from the

costly redesign and/or retraining of the InferNet. Thus, both the new and legacy mobiles with and without neural network masking, respectively, can coexist in using the cloud inference service. Depending on the remaining battery energy, a mobile can also opt in to or out of PriMask seamlessly without needing to inform the ISP. This kind of privacy preservation mechanism is called *cascadable*. **Fourth**, PriMask scales well with the number of mobiles using the inference service.

Now, the basic design to meet the first three objectives except scalability is described here. To implement the cascadable feature, the mobile applies a small-scale mask neural network (MaskNet) on the inference sample and then transmits the output to the ISP. The MaskNet can be obtained by training the concatenation of MaskNet and InferNet. During the training, the InferNet’s parameters are fixed and only the training loss is backpropagated from the InferNet to the MaskNet. Fig. 3.1(a) illustrates the cascadable feature achieved by MaskNet. To counteract the privacy threat from the curious ISP, the training is performed by a Privacy Service Provider (PSP) that is trusted by both the mobiles and the ISP. To keep the confidentiality of the ISP’s proprietary InferNet, *split learning* [70] is applied such that the MaskNet and InferNet are not revealed to ISP and PSP, respectively. To be collusion-resilient, independent split learning processes can be performed with distinct initialization seeds to yield heterogeneous MaskNets for mobiles. The heterogeneity is essential to collusion resilience, because otherwise the model inversion attack [68, 69] launched by the ISP using a colluding mobile’s MaskNet is effective to all mobiles using the same MaskNet. Fig. 3.1(b) illustrates the heterogeneity that provides the resilience against the potential collusion.

The above basic design requires a separate split training process for each mobile, rendering it non-scalable. For scalability, this chapter advances the design with inspiration from *HyperNet* [71], which is a generative neural network supervising the parameter updates of another neural network. Given random seeds, a HyperNet generates neural networks with identical architecture but distinct parameters for the same inference task. This is consistent with the heterogeneity requirement for collusion resilience. Thus, if the process of training a MaskNet in the basic design is replaced with inferencing a HyperNet that is much faster, PriMask becomes more scalable in terms of the generation speed of MaskNets. However, the approach needs to address two issues. First, different from the

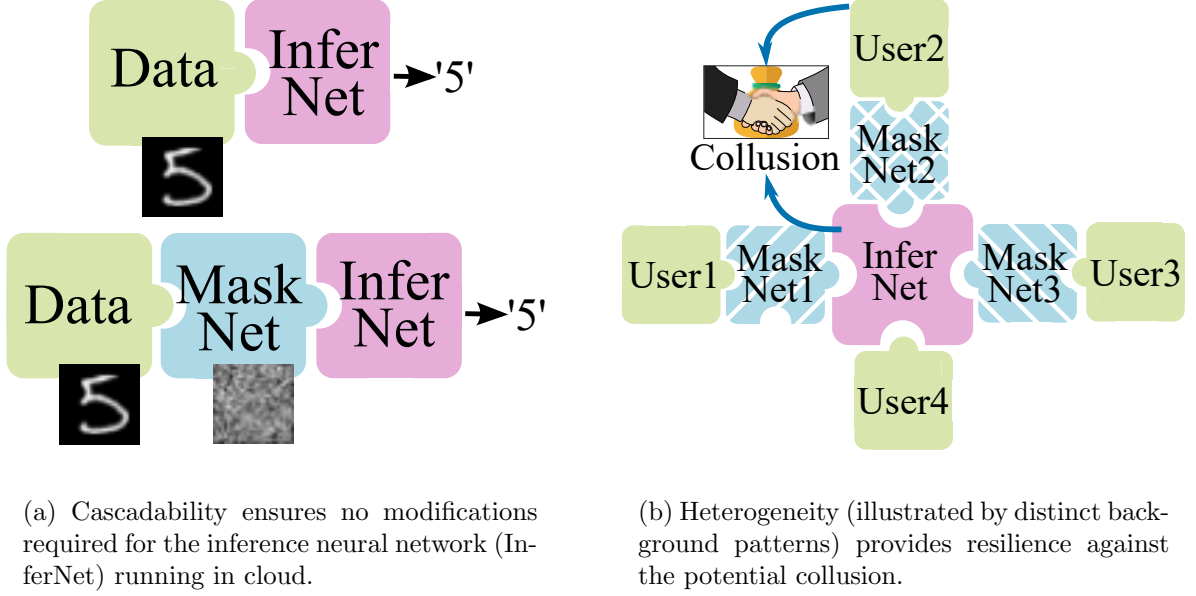


Fig. 3.1: Properties of heterogeneous MaskNets.

original concept of HyperNet that generates neural networks for a certain inference task, the approach needs a new design of HyperNet to generate MaskNets subject to the ISP's existing InferNet. Second, as HyperNet-generated MaskNets are correlated, the model inversion attack constructed against a specific MaskNet may be transferable to other MaskNets. To address these two issues, this chapter designs a *split adversarial learning* (SAL) method for training the HyperNet. Specifically, the PSP iteratively trains the HyperNet as defender and an attack neural network (AttackNet) as attacker that implements model inversion or private attribute extraction. During SAL, the HyperNet and InferNet are not revealed to the ISP and PSP, respectively. On the completion of SAL, it is difficult for any adversary who has obtained any HyperNet-generated MaskNet to construct an effective AttackNet.

This chapter's contributions are summarized as follows:

- Different from the existing neural network masking methods [63–67] addressing external eavesdroppers, PriMask counteracts curious ISP, mobile-ISP collusion, and requires no changes to the ISP's service.
- This chapter designs HyperNet and its SAL training method to improve PriMask's

scalability. Inferencing the HyperNet to generate a MaskNet takes only milliseconds up to two seconds on a workstation-class PSP.

- This chapter applies PriMask to three mobile sensing applications with diverse modalities and complexities, i.e., human activity recognition with inertial time series data, urban environment crowdsensing with one-shot tabular data, and driver behavior recognition with image. Evaluation shows PriMask’s privacy protection performance and scalability to support up to 100,000 mobiles.

For simplicity of exposition, this chapter assumes that the PSP is trusted by the mobiles. Understood in a different way, PriMask escalates the trustworthiness of the ISP to the level of the PSP, which is useful in practice. For instance, a major cloud computing service provider (e.g., Google) can act as the PSP to escalate the trustworthiness of its small-business tenants who provide inference services. As such, the mobile users can trust the inference services at the level that they trust the major cloud computing service provider.

Chapter organization: §3.2 states the problem. §3.3 presents the design of PriMask. §3.4, §3.5, and §3.6 present the three mobile sensing applications and evaluation. §3.7 discusses related issues. §3.8 concludes this chapter.

3.2 Problem Statement

3.2.1 System Overview and Threat Model

As illustrated in Fig. 3.2, this chapter considers a system consisting of a cloud-based Inference Service Provider (ISP), many mobile devices that desire to use the ISP’s service, and a Privacy Service Provider (PSP) that aims at enabling the mobiles to use the ISP’s service with certain privacy preserved. The privacy notion will be defined in §3.2.2. We assume that the ISP uses a deep neural network called InferNet to provide the inference service based on raw input data. InferNet can be large-scale and/or proprietary. Before PSP can serve the mobiles, it collaborates with the ISP by following a protocol to train a neural network called HyperNet. The details of the training approach and the protocol are in §3.3. The mobiles that do not desire privacy preservation can send the raw data to

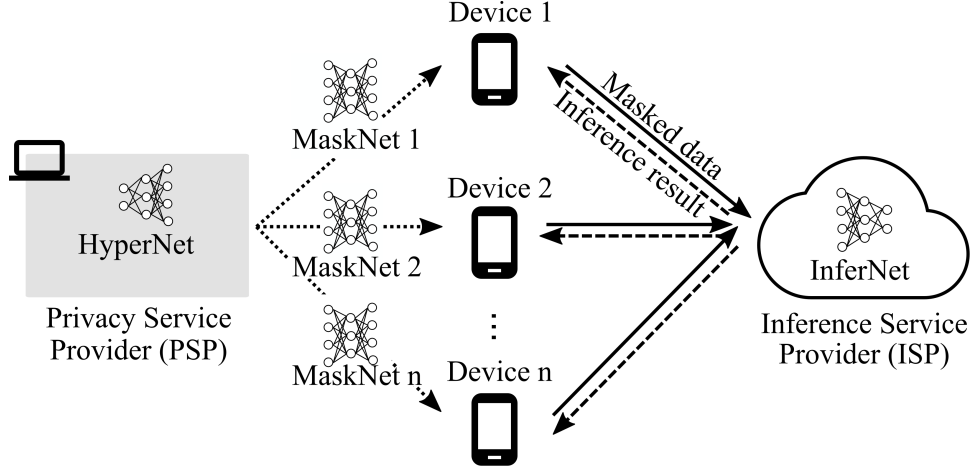


Fig. 3.2: System model.

ISP for inferencing InferNet. For each mobile that desires privacy preservation, the PSP infers the HyperNet with a random seed to generate a small-scale neural network called MaskNet and releases it to the mobile. The MaskNet has identical input and output dimensions. The MaskNets used by the mobiles are identical in architecture but heterogeneous in parameters. When a mobile desires privacy preservation, it feeds the raw data to its MaskNet and sends the output (i.e., masked data) to the ISP. Then, the ISP feeds the masked data to InferNet and returns the result to the mobile. Each mobile should keep its MaskNet confidential.

PriMask's threat model has three aspects:

- *Honest-but-curious ISP*: The ISP honestly executes the InferNet and does not tamper with the received data and the inference results. The ISP also honestly follows the protocol with the PSP to train the HyperNet. However, the ISP is curious about the private information contained in the data received from mobiles and aims at launching privacy attack.

- *Potential collusion between mobiles and ISP*: Some mobiles may collude with the ISP to find out other mobiles' privacy. Such colluding mobiles surrender their MaskNets to the ISP, which tries to launch privacy attack on non-colluding mobiles. In practice, the ISP may recruit such colluding mobiles by offering monetary benefits. The privacy preservation for such compromised mobiles becomes void. The case in which the ISP pretends mobiles to request MaskNets from the PSP is equivalent to colluding with mobiles.

■ *PSP trusted by mobiles and semi-trusted by ISP*: The PSP honestly performs its role aiming at ensuring the ISP’s quality of service while protecting the privacy of the mobiles. Thus, the PSP is trusted by the mobiles. However, it is only semi-trusted by the ISP in that the ISP cannot trustfully disclose its role-defining property of InferNet to the PSP.

PriMask aims at preserving the non-colluding mobiles’ privacy from the privacy attacks launched by the ISP solely or in collaboration with the colluding mobiles. PriMask does not regard the inference result as private information. CryptoNets [20] protects confidentiality of inference result since ISP can only obtain the homomorphically encrypted result. However, the homomorphic encryption of CryptoNets is still not practical for resource-constrained devices. CryptoNets is also not cascable and requires a redesign of the InferNet.

3.2.2 Privacy Notation and Attacks

Adversary goal: After receiving the masked inference samples from the non-colluding mobiles, the ISP aims at *either* reconstructing the original inference samples, *or* extracting a certain private attribute from each masked sample. These two adversary goals are referred to as *inversion attack* and *private attribute extraction*. The level of privacy protection can be measured by the average dissimilarity between the original and reconstructed samples and the accuracy of the extracted private attributes, respectively. This chapter adopts both mean squared error (MSE) and structural similarity index measure (SSIM) as the dissimilarity/similarity metric. Note that recent studies also consider the same privacy notions defined by inversion attack [65,66,68,69,72,73] and private attribute extraction [33,36,65,66].

Now, the implementations of the privacy attacks after the ISP obtains the MaskNet are discussed. The discussions also explain PriMask’s system model presented in §3.2.1.

■ *Inversion attack*: The study [68] presented an inversion attack approach using maximum likelihood estimation. Here this section describes a training-based approach. Specifically, the ISP feeds many samples to the MaskNet and obtains the outputs to form a training dataset. Then, the ISP trains a neural network called InvNet that estimates the MaskNet’s input from its output. The InvNet can use a mirrored architecture of the

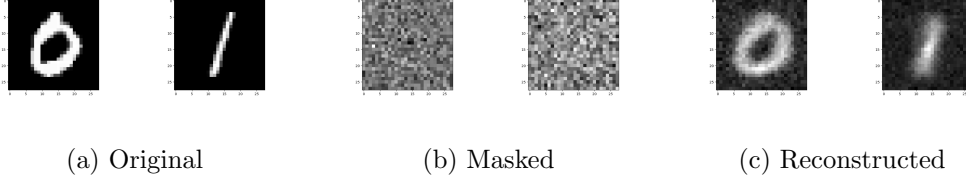


Fig. 3.3: Inversion attack on MNIST dataset.

MaskNet. Now, the effectiveness of the inversion attack using the MNIST handwritten digit dataset is shown in [74]. The MaskNet is generated using our approach described in §3.3. It is a two-layer multilayer perceptron (MLP). The InvNet with a mirrored architecture is trained with MSE of the inversion as the loss function.

Fig. 3.3 shows the original, masked, and reconstructed samples for two handwritten digits. The MaskNet can effectively mask the data. However, once the ISP obtains the used MaskNet, it can train the InvNet and reconstruct the original data to certain extents.

■ *Private attribute extraction:* Once the ISP obtains the MaskNet, it can also train a neural network called ExtNet to extract a certain private attribute from the masked data. Specifically, the ISP can feed many samples with private attribute labels to the MaskNet. The MaskNet’s outputs labeled with the corresponding private attributes form a training dataset that can be used to train ExtNet. As shown in our human activity recognition application (§3.4), once the ISP obtains the MaskNet, it can train the ExtNet to re-identify the user among 30 users with 63% accuracy.

The above results give the following implications. First, the generation and release of MaskNets should be performed by the PSP. An authority or a certified organization can be the PSP. Moreover, as discussed in §3.1, a major cloud computing service provider can be the PSP to pass the same trustworthiness on to its small-business tenants that provide inference services. Second, the mobiles’ MaskNets should not be identical. Otherwise, the ISP can launch effective privacy attacks on all mobiles once any one of them colludes with ISP. Third, the MaskNets and proprietary InferNet should be kept confidential to the ISP and PSP, respectively.

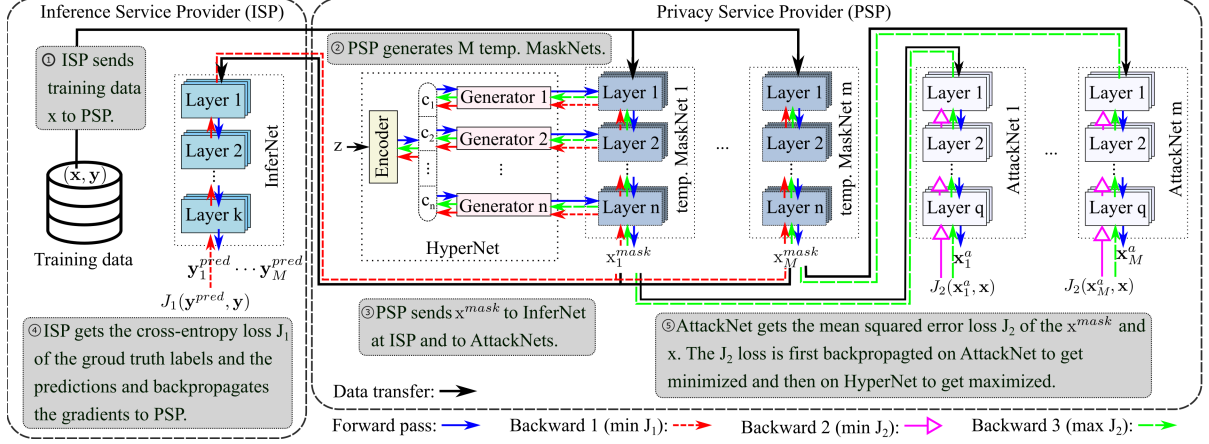


Fig. 3.4: Split adversarial learning (SAL) framework for training the HyperNet used to generate MaskNets. In SAL, the PSP needs unlabeled training data \mathbf{x} , which can be from an open dataset or the ISP as illustrated in the figure.

3.3 PriMask Design & Implementation

§3.3.1 presents the preliminary on HyperNet. §3.3.2 presents the split adversarial learning (SAL) framework for the HyperNet used to generate MaskNets. §3.3.3 presents the SAL protocol between the PSP and ISP. §3.3.4 discusses the generalizability and implementations of PriMask. §3.3.5 presents the results on MNIST as a simple case study.

3.3.1 Preliminary on HyperNet

HyperNet [71] is a neural network generating the parameters of the target neural network. This chapter uses HyperNet to generate MaskNets. As illustrated in Fig. 3.4, the HyperNet consists of an *encoder* network and n *weight generator* networks, where the HyperNet's parameters $\phi = \{\phi_E, \phi_G\}$, ϕ_E denotes the encoder's parameters, ϕ_G denotes all generators' parameters. The encoder takes as input a random vector \mathbf{z} sampled from a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The encoder maps \mathbf{z} to n latent codes denoted by $\{c_i | i = 1, 2, \dots, n\}$, which are then fed to n weight generators, respectively. Each weight generator outputs the parameters of a layer of the target neural network. The above process of inferencing the HyperNet, which is represented by $h(\mathbf{z}; \phi)$, can complete in a short time. By repeating the process with different inputs \mathbf{z} , many distinct neural

networks can be generated. The training of HyperNet is addressed in the following subsections.

3.3.2 Split Adversarial Learning Framework

Fig. 3.4 illustrates the SAL framework for training HyperNet to generate MaskNets. It integrates the principles of split learning [70] and adversarial learning [75]. It consists of four modules: InferNet at ISP, HyperNet and M AttackNets at PSP, and M temporary MaskNets generated by HyperNet. The M is a training hyperparameter. The AttackNets trained by the PSP form the adversary of the adversarial learning [75], which assists the PSP to train the HyperNet that can generate MaskNets more robust against the privacy attack launched by the ISP. Depending on the privacy protection goal (inversion attack or private attribute extraction), the AttackNet can be either InvNet or ExtNet. The core of SAL is the definitions of the training loss functions.

The notation used in this subsection is defined as follows. Denote by $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ the set of training samples, by $\mathbf{y} = \{y_1, y_2, \dots, y_N\}$ the corresponding class labels, and by $\mathbf{a} = \{a_1, a_2, \dots, a_N\}$ the corresponding private attribute labels, where N is the cardinality of the training dataset. Denote by $f_{\text{Mask}}(\cdot; \boldsymbol{\theta}_m)$ the m^{th} temporary MaskNet, where $\boldsymbol{\theta}_m$ represents the parameters generated by the HyperNet, i.e., $\boldsymbol{\theta}_m = h(\mathbf{z}_m; \boldsymbol{\phi})$. Denote by $f_{\text{Inf}}(\cdot; \boldsymbol{\psi})$ the pre-trained proprietary InferNet, where $\boldsymbol{\psi}$ represents the parameters that are constant during SAL. Denote by $f_{\text{Att}}(\cdot; \boldsymbol{\xi}_m)$ the m^{th} AttackNet, where $\boldsymbol{\xi}_m$ represents the parameters. During adversarial learning, the m^{th} AttackNet is used as the adversary against the m^{th} temporary MaskNet. Denote by $J(\mathbf{y}^{\text{pred}}, \mathbf{y})$ the cross-entropy loss function, where \mathbf{y} and \mathbf{y}^{pred} are the ground-truth and predicted labels. The cross-entropy loss function also admits privacy labels, i.e., $J(\mathbf{a}^{\text{pred}}, \mathbf{a})$. Denote by $E(\mathbf{x}^a, \mathbf{x})$ the MSE loss, where \mathbf{x} and \mathbf{x}^a are the original samples and those reconstructed by InvNet.

Now, in this section, the loss functions are defined here. When a batch of M HyperNet-generated MaskNets are used, the ISP's quality of service is characterized by the following cross-entropy loss:

$$J_1 = \frac{1}{M} \sum_{m=1}^M J(f_{\text{Inf}}(f_{\text{Mask}}(\mathbf{x}; h(\mathbf{z}_m; \boldsymbol{\phi})); \boldsymbol{\psi}), \mathbf{y}). \quad (3.1)$$

Depending on the privacy protection goal, effectiveness of PSP's m^{th} AttackNet is characterized by the following loss:

$$J_{2,m}(\xi_m) = \begin{cases} E(f_{\text{Att}}(f_{\text{Mask}}(\mathbf{x}; h(\mathbf{z}_m; \phi)); \xi_m), \mathbf{x}), & \text{for InvNet;} \\ J(f_{\text{Att}}(f_{\text{Mask}}(\mathbf{x}; h(\mathbf{z}_m; \phi)); \xi_m), \mathbf{a}), & \text{for ExtNet.} \end{cases}$$

The SAL flow is as follows. First, HyperNet is trained to minimize J_1 , i.e., $\phi^* = \arg \min_{\phi} \mathbb{E}_{\mathbf{z}_m \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \forall m \in [1, M]} [J_1]$. Then, each AttackNet is trained against the corresponding temporary HyperNet-generated MaskNet to minimize $J_{2,m}$: $\xi_m^* = \arg \min_{\xi_m} J_{2,m}(\xi_m)$, $\forall m \in [1, M]$. Lastly, the HyperNet is refined to achieve a multi-objective goal of minimizing J_1 and maximizing each $J_{2,m}(\xi_m^*)$, where the latter aims at defeating the privacy attack. This section represents the multi-objective goal using a single composite loss function to direct the refinement of the HyperNet:

$$\phi^{**} = \arg \min_{\phi} \mathbb{E}_{\mathbf{z}_m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[J_1 - \frac{\lambda}{M} \sum_{m=1}^M J_{2,m}(\xi_m^*) \right], \quad (3.2)$$

where the *adversarial learning factor* λ balances the objectives of maintaining the ISP's quality of service and defeating the privacy attack. The λ can be used to tune the trade-off between the inference service quality and the privacy protection level. In §3.3.3, this section will discuss how to set λ .

3.3.3 Split Adversarial Learning Protocol

This section presents the protocol between ISP and PSP to implement SAL. To drive SAL, the PSP needs to feed unlabeled data samples \mathbf{X} from the training dataset (\mathbf{X}, \mathbf{Y}) to the HyperNet-generated MaskNets. If the training dataset is not publicly available, the ISP transmits \mathbf{X} (excluding \mathbf{Y}) to the PSP, as illustrated by step ① in Fig. 3.4. Then, ISP and PSP start training. In each loop of a training epoch, the SAL protocol has the following three phases.

(1) Updating HyperNet: The PSP samples a mini-batch \mathbf{x} from \mathbf{X} . On \mathbf{x} , the PSP draws M random vectors $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_M$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and generates M MaskNets using the current HyperNet, as illustrated by step ② in Fig. 3.4. The masked mini-batch by the m^{th} MaskNet is denoted by $\mathbf{x}_m^{\text{mask}}$. To compute the gradient of Eq. (3.1), the PSP sends $\{\mathbf{x}_1^{\text{mask}}, \mathbf{x}_2^{\text{mask}}, \dots, \mathbf{x}_m^{\text{mask}}\}$ to the ISP, as illustrated by step ③ in Fig. 3.4. The ISP feeds the

received masked data to the InferNet to obtain the predictions $\{\mathbf{y}_1^{\text{pred}}, \mathbf{y}_2^{\text{pred}}, \dots, \mathbf{y}_M^{\text{pred}}\}$. Then, the ISP computes the cross-entropy loss J_1 for the mini-batch using the ground truth labels \mathbf{y} and sends the backward gradients of $\{\mathbf{x}_1^{\text{mask}}, \mathbf{x}_2^{\text{mask}}, \dots, \mathbf{x}_m^{\text{mask}}\}$ to the PSP, as illustrated by step ④ in Fig. 3.4. The InferNet’s parameters remain unchanged during SAL; they are solely used to compute the backward gradients. Upon receiving the backward gradients, the PSP backpropagates them through the MaskNets without updating their parameters and then through the HyperNet to update its parameters ϕ for minimizing J_1 .

(2) Updating AttackNets: After updating ϕ on \mathbf{x} , PSP enters the adversarial learning phase to update the M AttackNets. Specifically, the PSP regenerates M MaskNets using newly sampled random vectors $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_M\}$ and the latest ϕ . The masked mini-batch $\mathbf{x}_m^{\text{mask}}$ produced by the m^{th} updated MaskNet is fed into the corresponding AttackNet, as illustrated by step ③ in Fig. 3.4. The MSE/cross-entropy loss is backpropagated to update the AttackNets’ parameters $\{\xi_1, \xi_2, \dots, \xi_M\}$ to minimize $J_{2,m}$, as illustrated by step ⑤ in Fig. 3.4. For the same mini-match, the PSP repeats the above process for multiple times to gain better AttackNets.

(3) Refining HyperNet: The last phase on the current mini-batch \mathbf{x} is to refine the HyperNet according to the composite loss function in Eq. (3.2). Similar to Phase (1), the PSP sends the latest $\{\mathbf{x}_1^{\text{mask}}, \mathbf{x}_2^{\text{mask}}, \dots, \mathbf{x}_m^{\text{mask}}\}$ to the ISP and receives the backward gradients corresponding to the loss J_1 . The PSP also computes the backward gradients of the AttackNets corresponding to the losses $\{J_{2,1}, J_{2,2}, \dots, J_{2,M}\}$. PSP updates HyperNet’s parameters ϕ according to Eq. (3.2).

The PSP repeats the above three phases on multiple mini-batches in the current training epoch. Once all the training samples are utilized in the current epoch, the PSP proceeds to the next epoch. Upon the completion of the training, the PSP is ready to serve the mobiles. Specifically, to respond to a mobile’s service request, the PSP feeds a random vector \mathbf{z} sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to the HyperNet to generate a MaskNet and releases it to the mobile.

Now, this section discusses the setting of the adversarial learning factor λ . The PSP may perform SAL for multiple rounds with different λ settings to train multiple HyperNets. For each SAL process, the PSP may measure the average test accuracy and

Table 3.1: PriMask applications and benchmark results including model sizes and compute times (unit: ms).

Application	MNIST	HAR	UEC	DBR
Sensor	camera	IMU	multiple*	camera
Data type	28x28 image	time series	one-shot tabular	240x240 image
Sample size	784	1,152	5	57,600
Private attribute	n/a	identity	location	identity
InferNet size (MB)	0.08	8.35	0.06	226.37
HyperNet size (MB)	12.64	113.01	0.20	3310
MaskNet size (MB)	0.20	1.76	0.003	54
MaskNet generation [†]	1.5	2.3	1.6	2130
MaskNet execution [‡]	0.011	1.33	0.03	42.02
MaskNet execution [*]	0.078	1.14	0.05	21.24

*Light, microphone, air pressure, temperature.

[†]On a computer with an i7-6850K CPU and a Quadro RTX 6000 GPU.

[‡]On Jetson Nano’s quad-core Cortex-A57 processor.

^{*}On Google Pixels 4’s octa-core Qualcomm Snapdragon 855 processor.

the metric characterizing the effectiveness of the privacy attack (e.g., the average MSE of the inversion attack on non-colluding mobiles). The PSP can publish a table of λ settings and the associated test accuracy and privacy attack effectiveness metric. Each mobile may inform the PSP with its preferred λ setting and obtain a MaskNet generated by the PSP using the corresponding HyperNet.

3.3.4 Generalizability and Implementations

As SAL is agnostic to InferNet, PriMask can be applied to different inference tasks. In §3.3.5, PriMask is applied to a simple handwritten digit recognition task as a starting case study. In §3.4, §3.5, and §3.6, PriMask is applied to three mobile sensing tasks of human activity recognition (HAR), urban environment crowdsensing (UEC), and driver behavior recognition (DBR). As summarized in Table 3.1, the three applications have diverse sensing modalities, data types, and InferNet complexities. All these four applications use similar MaskNet and HyperNet architectures, with minor differences in the numbers of neurons in the layers to be compatible with the dimensions of the input/output data. Therefore, PriMask has good generalizability.

PyTorch [76] is used to implement the InferNets and HyperNets on workstation computers. PyTorch and PyTorch Mobile [77] are used to implement the MaskNet-based data masking on Jetson Nano [78] running Ubuntu OS and Google Pixel4 smartphone running Android OS, respectively.

Table 3.1 shows the sizes of the models used by PriMask and the compute overhead measurements for all the four applications. The compute times in Table 3.1 are average values over 1,000 executions. For each application, the size of HyperNet is larger than the size of InferNet. As HyperNets are executed on PSP’s server-class computers, their large sizes are not a concern. The time for executing HyperNet to generate a MaskNet is at most 2.13 seconds. Executing MaskNet to mask a sample on Jetson Nano and Pixel4 just takes tens of milliseconds at most, representing low overheads.

3.3.5 A Simple Case Study on MNIST

As the MNIST dataset facilitates visualization, it is used as a starting and simple case study. MNIST consists of 60,000 training samples and 10,000 testing samples. Each sample is a 28×28 image showing a handwritten digit. The pixel value is normalized to $[0, 1]$. A convolutional neural network (CNN) is designed as the InferNet. The CNN consists of two convolutional layers with max pooling, three dense layers with Rectified Linear Unit (ReLU) activation, and a softmax function to generate the classification result. The test accuracy of the InferNet on raw testing samples is 98.7%. The MaskNet adopts an MLP architecture with a single hidden layer consisting of 32 neurons. There are 25,120 trainable parameters between the input and hidden layers, and 25,872 trainable parameters between the hidden and output layers. Thus, HyperNet’s output consists of two parts corresponding to the above two groups of parameters. Fig. 3.5 shows the HyperNet’s architecture.

The training samples including labels are used by ISP to build InferNet. The training samples excluding labels are used by PSP to build HyperNet in collaboration with ISP according to the SAL protocol. To evaluate a MaskNet generated by HyperNet, all test samples are fed to the MaskNet-InferNet pipeline and measure the test accuracy. Since our focus is to understand the impact of MaskNet on the inference, there is no need to

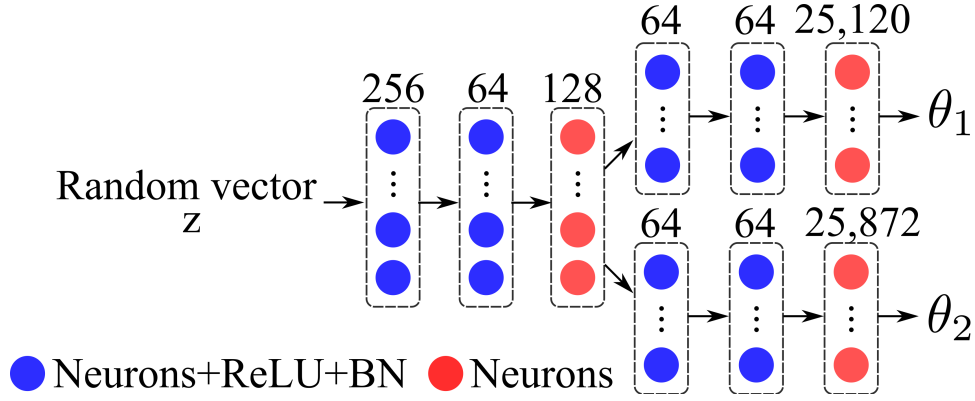


Fig. 3.5: HyperNet architecture.

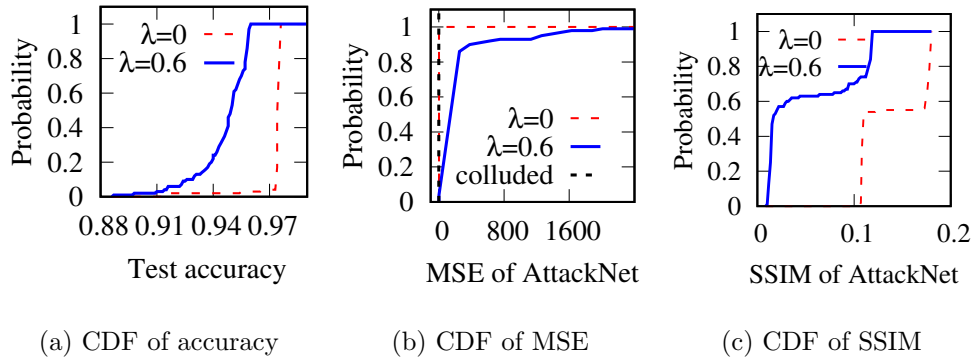


Fig. 3.6: Impact of PriMask on MNIST test accuracy and privacy protection. λ is adversarial learning factor ($\lambda = 0$ means that adversarial learning is not enabled).

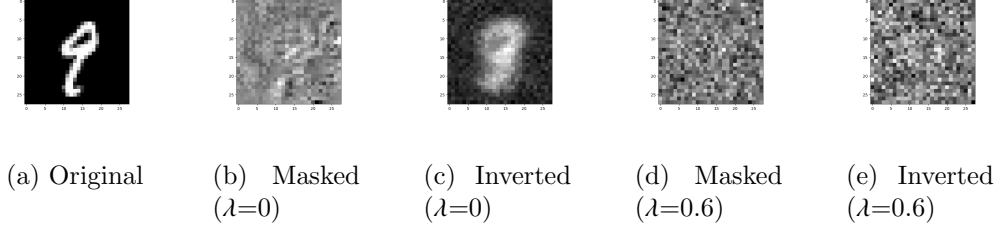


Fig. 3.7: Original, masked, reconstructed samples. (c) and (e) show ISP’s reconstructions with the smallest MSEs when adversarial learning factor λ is 0 and 0.6.

simulate the system by assigning disjoint portions of the test dataset to all mobiles. This evaluation methodology is followed throughout this chapter.

Impact of PriMask on InferNet accuracy: The HyperNet is used to generate MaskNets for 100 mobiles. First, the InferNet’s test accuracies are evaluated when the 100 MaskNets are used. Fig. 3.6(a) shows the cumulative distribution functions (CDFs) of test accuracies when the adversarial learning factor $\lambda = 0$ and $\lambda = 0.6$. When $\lambda = 0$, the adversarial learning of SAL is not enabled. In this case, the InferNet’s test accuracies corresponding to the 100 MaskNets are mostly within (95.5%, 97.6%), with an average value of 97.2%. When $\lambda = 0.6$, the test accuracies are mostly within (91.5%, 95.9%), with an average value of 94.5%. Compared with the original test accuracy of 98.7%, PriMask results in average test accuracy losses of 1.5% and 4.2%, when the adversarial learning is disabled and enabled, respectively. This chapter will show shortly that the adversarial learning enhances the privacy protection. Therefore, there is a trade-off between maintaining test accuracy and preserving privacy.

Resilience against mobile-ISP collusion: For this MNIST example, this chapter only consider the privacy threat of inversion attack. Suppose one of the 100 mobiles colludes with ISP. The MSEs and SSIMs of the inversion attack on the non-colluding mobiles are measured. MSE and SSIM are complementary in characterizing the privacy loss caused by the inversion attack. MSE measures the average pixel-wise difference between the original and reconstructed samples. However, MSE falls short of characterizing their correlation. SSIM, which is a perceptual metric quantifying quality degradation of reconstructed image, captures the correlation. Fig. 3.6(b) shows the CDFs of the MSEs when $\lambda = 0$ and $\lambda = 0.6$. The vertical line in Fig. 3.6(b) shows the inversion MSE for the colluding

mobile when $\lambda = 0$, which is 0.71. The inversion MSEs for the non-colluding mobiles are distributed from 0.76 to 1.25, larger than that for the colluding mobile. When adversarial learning is not adopted (i.e., $\lambda = 0$), the inversion MSEs for all the non-colluding mobiles are higher than that of the colluding mobile. When adversarial learning is adopted with $\lambda = 0.6$, the inversion MSEs of the non-colluding mobiles are dispersed in a much wider range of 1.77 to 2,630, larger than the MSEs when $\lambda = 0$. Fig. 3.6(c) shows the CDFs of SSIMs when $\lambda = 0$ and $\lambda = 0.6$. Note that the maximum value of SSIM is 1, indicating the highest structural similarity. From Fig. 3.6(c), the SSIMs when $\lambda = 0.6$ are smaller than those when $\lambda = 0$, suggesting that adversarial learning reduces the structural similarity. Fig. 3.7 shows the original and masked samples and the ISP’s inversion results with the smallest MSE for a non-colluding mobile when $\lambda = 0$ and $\lambda = 0.6$. These samples show that the adversarial learning is effective in strengthening the privacy protection against the collusion-based inversion attack.

The existing neural network masking approaches [63–67] designed to address external eavesdroppers are vulnerable to the inversion attack and private attribute extraction after the curious ISP obtains the MaskNet via collusion. The effect of inversion as a result of collusion has been shown in §3.2.2. Due to the basic difference in the resilience against collusion, this chapter does not compare PriMask with these neural network masking approaches in evaluation.

3.4 Human Activity Recognition

Human activity recognition (HAR) with the data from the inertial measurement units (IMUs) of a user’s mobile is a basic building block of mobile sensing applications for human-computer interaction, human behavior characterization, and smart health. However, IMU data may contain private information related to identity, gender, and age [79, 80]. In this section, PriMask is applied to an HAR system to counteract both the inversion attack and private attribute extraction.

3.4.1 HAR Dataset, InferNet, and HyperNet

A public dataset [81] is used which is collected from 30 human volunteers performing six types of daily activities (walking, walking upstairs, walking downstairs, sitting, stand-

ing, and laying). Each volunteer carried a waist-mounted smartphone for recording the accelerometer and gyroscope data. The recorded data include 3-axial linear acceleration with/without gravity and 3-axial angular velocity sampled at 50 sps. Thus, each record has nine components. The record traces are pre-processed by noise filters and then arranged in sliding windows of 2.56 seconds with 50% overlap. The trace within a window is referred to as a *data sample*. Thus, each data sample is a tensor sized $9 \times 1 \times 128$. Each data sample has an activity label. The dataset contains 10,299 data samples that are partitioned into the training and testing subsets by 7:3. Each data sample also has a volunteer identity label to indicate which volunteer that the sample was collected from. The identity is regarded as the private attribute.

A CNN InferNet is designed, consisting of two convolutional layers with max pooling, three dense layers with ReLU activation, and softmax function. The first convolutional layer admits a 1,152-dimensional vector flattened from the data sample tensor and applies 32 1×9 convolution filters. The second convolutional layer applies 64 1×9 filters. The three dense layers have 1,000, 500, and 6 neurons. To avoid overfitting, dropout is adopted on dense layers. The test accuracy of the trained InferNet on raw data samples is 92.5%.

The MaskNet adopts a two-layer MLP architecture. For both the input and output layers, the number of neurons is 1,152. The middle layer has 200 neurons. There are 230,600 trainable parameters between the input and middle layers, and 231,552 between the middle and output layers. The HyperNet adopts a similar architecture as shown in Fig. 3.5, with slight modifications on the number of the neurons.

3.4.2 Evaluation Results for HAR

Three HyperNets are trained. For the first HyperNet, adversarial learning is disabled (i.e., $\lambda = 0$). Therefore, this HyperNet is agnostic to the type of privacy attack. The other two HyperNets are adversarially trained to counteract the inversion attack (with $\lambda = 0.3$) and private attribute extraction (with $\lambda = 0.1$), respectively. By default, each HyperNet is used to generate 100 MaskNets for evaluation. 100,000 MaskNets are also generated to evaluate scalability of PriMask.

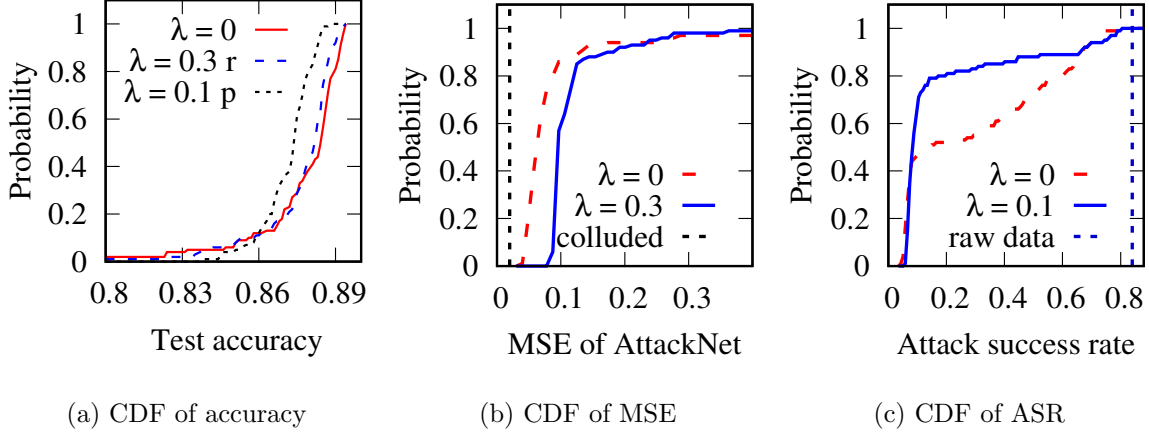


Fig. 3.8: Impact of PriMask on HAR test accuracy and privacy protection. Legends denoted by ‘r’ and ‘p’ are for HyperNets adversarially trained with inversion attack and private attribute extraction, respectively.

Table 3.2: Statistics of InferNet’s test accuracies across 100,000 MaskNets ($\lambda = 0.1$; AttackNet = ExtNet).

Post-generation validation	test accuracy range	mean
Not applied	(0.52, 0.90)	0.87
Applied	(0.80, 0.90)	0.88

3.4.2.1 Impact of PriMask on InferNet accuracy

Fig. 3.8(a) shows the CDF of the InferNet’s test accuracy corresponding to the three HyperNets. The test accuracies are distributed within (80%, 90%). The average test accuracies for the three CDFs are 87.6%, 87.0%, 87.5%. Therefore, on average, there are accuracy drops of 5% to 5.5%.

This section further evaluates PriMask’s scalability in terms of InferNet accuracy. 100,000 MaskNets are generated using the HyperNet adversarially trained against private attribute extraction with $\lambda = 0.1$. The first row of Table 3.2 summarizes InferNet’s test accuracies across all MaskNets. The mean value of the test accuracies (i.e., 87%) remains at the same level as in Fig. 3.8(a). This result is supportive of PriMask’s scalability in terms of InferNet accuracy. However, a few MaskNets lead to low InferNet accuracy of down to 52%. Less than 2% of all the MaskNets lead to InferNet accuracy lower than 80%. This suggests a long-tail distribution of InferNet’s accuracies across the MaskNets.

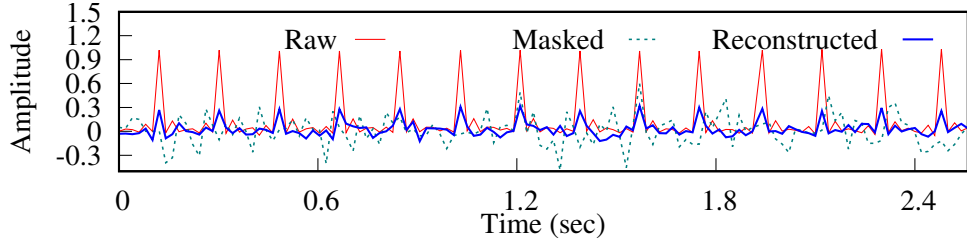
To avoid outlier MaskNets, a post-generation validation process is applied. Specifically, for each generated MaskNet, the PSP uses a small validation dataset and works with the ISP to measure the InferNet’s test accuracy. The PSP regenerates the MaskNet until the InferNet’s test accuracy exceeds a *passing threshold*. Only the validated MaskNets are released to mobiles. The second row of Table 3.2 summarizes the results if the validation is applied, where PSP’s validation dataset includes 100 samples and passing threshold is 80%. The average test accuracy increases to 88%. In addition, outlier MaskNets are not released.

3.4.2.2 Resilience against a single colluding mobile

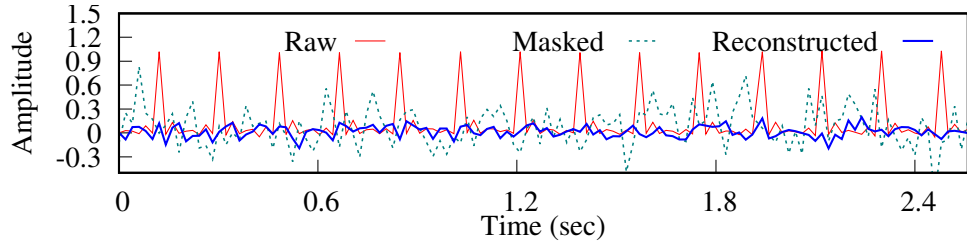
A system of 100 mobiles is considered and one of them colludes with ISP. Fig. 3.8(b) shows the CDFs of MSEs achieved by InvNet for non-colluding mobiles when the HyperNet is trained without or with adversarial learning. The vertical line represents the inversion MSE (i.e., 0.02) for the colluding mobile when $\lambda = 0$. The MSEs for non-colluding mobiles are higher than that for the colluding mobile. In addition, when adversarial learning is applied, MSEs are larger.

Fig. 3.8(c) shows the CDF of the attack success rate (ASR), i.e., the accuracy of the extracted private attribute, achieved by ExtNet on raw IMU data and masked data from non-colluding mobiles. The two CDFs are results for the HyperNets with and without adversarial learning against ExtNet. The vertical line represents ASR on raw IMU data (i.e., 84%). Note that since the data samples are collected from 30 volunteers [81], the random guessing strategy yields an ASR of $1/30 = 3.3\%$. The 84% ASR suggests that the IMU data contains abundant information regarding user identity. When adversarial learning is applied, the CDF is higher than that without adversarial learning. This shows that adversarial learning is effective in reducing ASR. The average ASRs with and without adversarial learning are 28.3% and 17%, respectively. Although MaskNets under the setting of $\lambda = 0.1$ cannot reduce ASR to the random guessing level, they have already achieved significant reductions in ASR compared with the case without private attribute protection. By setting larger λ , the ASRs for non-colluding mobiles will further decrease. But InferNet’s accuracy will decrease too.

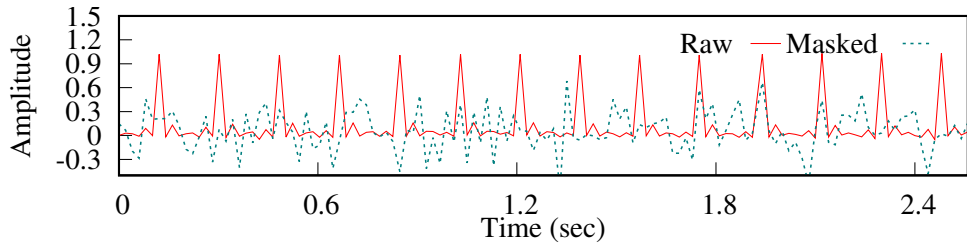
The three subfigures of Fig. 3.9 show the traces of the raw and ISP’s reconstructed data for the first axis of linear acceleration, and the first dimension of the masked data,



(a) HyperNet trained without adversarial learning, i.e., $\lambda = 0$



(b) HyperNet trained against inversion attack ($\lambda = 0.3$)



(c) HyperNet trained against private attribute extraction ($\lambda = 0.1$)

Fig. 3.9: Traces of raw and ISP's reconstructed data for the first axis of linear acceleration, as well as the first dimension of masked data for a certain non-colluding mobile. The ground truth human activity is standing.

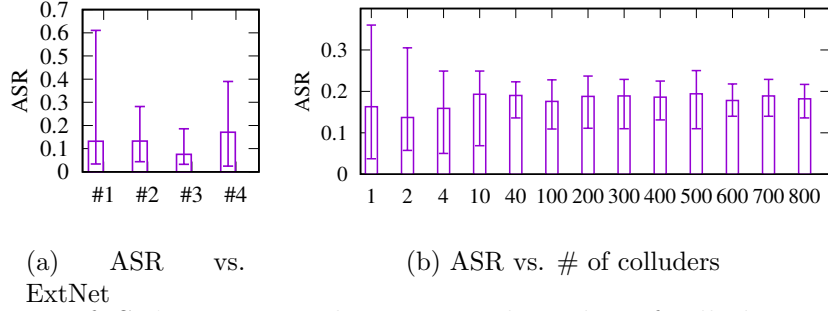


Fig. 3.10: Impact of ISP's ExtNet architecture and number of colluding mobiles on ASR. In (a), ExtNet#1 is also the architecture used by PSP in SAL. Whiskers of an error bar represent maximum and minimum.

for a certain non-colluding mobile adopting the three HyperNets, respectively. For linear acceleration, the peaks are salient features. From Fig. 3.9, the masked traces do not have salient peaks. In Fig. 3.9(a), without adversarial learning, the reconstructed trace still pronounces peaks at the same times of the original peaks. In Fig. 3.9(b), with adversarial learning, the reconstructed trace no longer pronounces peaks at the same times of the original peaks, suggesting better protection against inversion attack.

Next, the impact of the ExtNet architecture used by ISP is investigated on ASR of private attribute extraction. A system of 1,000 mobiles is considered and one of them colludes with ISP. The HyperNet adversarially trained against ExtNet with $\lambda = 0.1$ is used to generate MaskNets. Four ExtNet architectures are designed to be used by ISP, which are illustrated as:

- *ExtNet#1*: $C_{32}-C_{64}-D_{1664}-D_{1000}-D_{500}-D_{30}-\text{softmax}$
- *ExtNet#2*: $C_{32}-C_{64}-C_{128}-D_{1152}-D_{500}-D_{30}-\text{softmax}$
- *ExtNet#3*: $D_{1152}-D_{100}-D_{50}-D_{30}-\text{softmax}$
- *ExtNet#4*: $D_{1152}-D_{500}-D_{100}-D_{30}-\text{softmax}$

where C_n represents a convolutional layer with n filters followed by max pooling, D_n represents a dense layer of n neurons with ReLU activation. The ExtNet#1 architecture is identical to that used by the PSP's adversarial learning. In Fig. 3.10(a), each error bar shows the average, maximum, minimum of the ASRs across the 999 non-colluding mobiles when ISP adopts a certain ExtNet architecture. No ExtNet architecture shows clear advantage for ISP in terms of average ASR. This suggests that the resilience against collusion is insensitive to the ExtNet architecture used by the ISP.

3.4.2.3 Resilience against multiple colluding mobiles

A system of 1,000 mobiles is considered and the number of mobiles colluding with ISP varies. For a certain set of colluding mobiles, to build the training dataset for constructing ExtNet, ISP feeds each original training sample to all colluding mobiles' MaskNets to obtain multiple samples for training ExtNet. Thus, the ISP's trained ExtNet is expected to address all colluding mobiles' MaskNets. Fig. 3.10(b) shows the error bars of non-colluding mobiles' ASRs versus the number of colluding mobiles. While the average ASR exhibits an increasing trend when the number of colluding mobiles is lower than 10, it becomes flat at around 20% when the number of colluding mobiles is up to 800. Recall that, without PriMask's protection, ASR is up to 84% (cf. §3.4.2.2). The above results suggest that PriMask is resilient to increase of colluding mobiles. Note that the range of ASR shows a decreasing trend with the number of colluding mobiles. This is partly due to decreasing number of non-colluding mobiles for generating the ASR statistics (i.e., minimum/maximum).

3.5 Urban Environment Crowdsensing

The data collected in a city-wide experiment that involves over 10,000 school students is studied in our city to understand urban environment in a crowdsensing manner.¹ In the experiment, each participant carries a wearable device on a neck lanyard which integrates several sensors to record the surrounding environment conditions of the participant. The sensor measurements are transmitted opportunistically to a central cloud portal through over 18,000 Wi-Fi hotspots deployed across our city. On this crowdsensing platform, an urban environment crowdsensing (UEC) application is developed that classifies the ambient conditions of participants and identifies those which may result in discomfort for human beings. The potential of location leakage are investigated and apply PriMask to protect participants' location privacy in such a circumstance.

¹Ethical approvals for the city-wide experiment were obtained. This work only uses anonymized data from the experiment.

3.5.1 Dataset, InferNet, ExtNet, and PriMask

The dataset consists of 640,000 samples. Each sample includes five sensor readings of light intensity, noise level, atmospheric pressure, ambient temperature, and body-reflected temperature. Each sample is geotagged using the location information inferred from the wearable device’s nearby Wi-Fi hotspots. To build the UEC application, the five sensor readings are used. To investigate the potential location privacy leakage, the correlation is studied between the five sensor readings and the geotags. The city-wide experiment collected geotags for ground truth only. After the privacy-preserving UEC application is developed and deployed, geotags are no longer needed.

The wearable device used in the experiment does not support the participants to key in their real-time comfort feelings. In fact, requesting the participants to continuously or frequently provide feedback is impractical. Thus, several first principles are followed to generate the ground truth information regarding the discomfort level of the environment condition. The procedure is as follows. First, each sensor reading is normalized to $[0, 1]$. Then, the following three scoring functions is applied on the normalized sensor readings to generate discomfort scores: $d_1(x) = x$, $d_2(x) = 1 - x$, and $d_3(x) = 4 \cdot (x - 0.5)^2$. As human discomfort in general increases with noise level, $d_1(x)$ is applied to score noise level. In the climate zone of our city, as low atmospheric pressure is in general positively correlated with human discomfort, $d_2(x)$ is applied to score atmospheric pressure. As the light intensity, ambient temperature, and body-reflected temperature should be in their respective proper ranges, the quadratic function $d_3(x)$ is applied with the minimum (i.e., the least discomfort) at $x = 0.5$ on them.

Lastly, the five scores is sum up to obtain a final score to characterize the overall human discomfort. The users with high discomfort scores may need attention and preventative actions.

The private attribute labels are generated as follows. First, the k -means algorithm is applied with $k = 10$ to cluster the data’s geotags into ten zones. The zone ID is viewed as the private attribute. However, the numbers of samples in the clusters are imbalanced. To simplify the presentation of the evaluation results, the data is resampled to ensure class balance. The re-sampling generates a training dataset of 60,000 samples (i.e., 6,000

samples in each zone) and a testing dataset of 4,481 samples (i.e., about 448 samples in each zone).

Transmitting the participants' data to the cloud portal is preferred, because it supports various posterior data analytics including UEC. If raw data is transmitted, the scoring functions can be applied directly. However, to admit masked data, a regression InferNet is needed to approximate the sum of the scoring functions. An MLP is designed with three hidden layers as the InferNet, which have 10, 20, and 10 neurons with ReLU activation, respectively. The output layer is a single neuron giving the predicted discomfort score. The test accuracy is used to characterize the performance of the InferNet. Specifically, if the difference between the predicted score and the ground truth is less than 0.5 (i.e., 10% of the maximum discomfort score), the prediction is regarded as correct. The trained InferNet achieves a test accuracy of 99.2% on the raw testing data.

Then, an ExtNet is trained using the training data and the associated zone IDs. The ExtNet has two hidden layers, with 200 and 50 neurons using ReLU activation, respectively. It has an output layer with 10 neurons corresponding to the 10 zones. On the raw testing data, the ExtNet's ASR is 42%. As the strategy of randomly guessing the zone has 10% ASR only, the ExtNet's 42% ASR suggests that the sensor readings leak information regarding the participants' locations.

The MaskNet is an MLP with a single hidden layer. The input, hidden, and output layers have five neurons each. The MaskNet has 60 trainable parameters. The HyperNet adopts a similar architecture as shown in Fig. 3.5, with minor modifications on the number of neurons for MaskNet compatibility.

3.5.2 Evaluation Results for UEC

Three HyperNets are trained with the following settings: 1) $\lambda = 0$; 2) $\lambda = 0.2$ against inversion attack; 3) $\lambda = 0.1$ against private attribute extraction. Fig. 3.11(a) shows the CDFs of the test accuracies when the three HyperNets are used. The average test accuracy for the first and the second HyperNets are 98.3% and 96.2%, respectively. Thus, compared with the test accuracy obtained on raw testing data (i.e., 99.2%), there are 0.9% and 3% accuracy drops. When the third HyperNet is used, the average test accuracy drops to 80.1%. This is because $\lambda = 0.1$ is an aggressive setting against private attribute

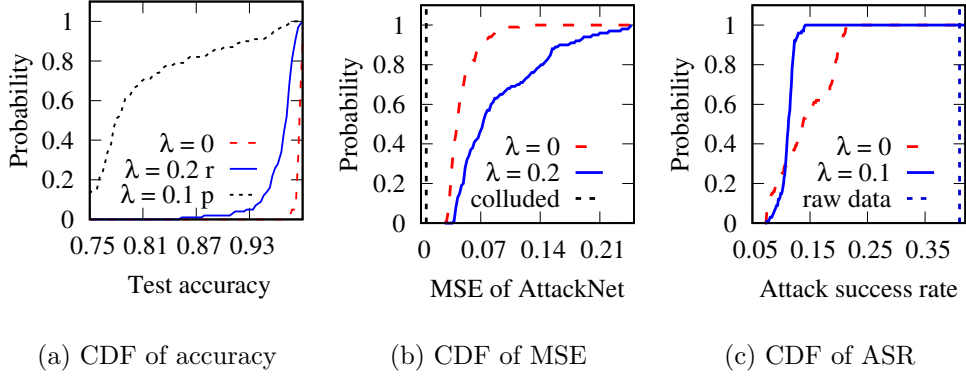


Fig. 3.11: Impact of PriMask on UEC test accuracy and privacy protection. Legends denoted by ‘r’ and ‘p’ are for HyperNets adversarially trained with inversion attack and privacy extraction.

extraction. This issue is presented along with the achieved privacy protection strength shortly.

Suppose a mobile colludes with ISP. Fig. 3.11(b) shows the CDFs of inversion MSEs when the first and the second HyperNets are used. The adversarial learning improves resilience against inversion attack in the presence of collusion. Fig. 3.11(c) shows the CDFs of ASRs when the first and the third HyperNets are used. When no adversarial learning (i.e., $\lambda = 0$) is applied, PriMask reduces ASR from the original 42% to 14.2% on average. As the 14.2% ASR is close to its lower bound of 10%, intuitively, more data utility will be sacrificed to further reduce ASR. Thus, when $\lambda = 0.1$ that produces an average ASR of 11%, significant test accuracy drops is observed in Fig. 3.11(a). Smaller λ settings can restore the test accuracy, while the resulting average ASRs will be within (11%, 14.2%), which are satisfactory.

3.6 Driver Behavior Recognition

In U.S., one in five car accidents is caused by a distracted driver [82]. Thus, using smartphone to detect driver’s engagement in distracted behaviors is useful. To incentivize drivers’ participation, the car insurance companies may provide premium discounts according to the monitoring results. To facilitate the design of driver behavior recognition (DBR), an insurance company initiated a competition [83] by providing a dataset of

images captured in cars regarding drivers’ behaviors. In this section, a CNN is trained based on the dataset . From our implementation, the CNN is heavy (226 MB) and inappropriate for local execution on phones. If it runs in the cloud, transmitting the raw images to the cloud inevitably incurs privacy concerns. Thus, in this section, PriMask is applied to design privacy-preserving cloud-based DBR.

3.6.1 DBR Dataset and System Design

The dataset consists of 22,424 grayscale images, each sized 240×240 . Each sample has a driver behavior label (in 10 classes) and driver identity as a private attribute label (in 26 classes). Examples of the driver behavior include safe driving, drinking, etc. The dataset is partitioned into training samples and testing samples by following a ratio of 8:2.

DBR is a complex task. A 32-layer CNN architecture described in [84] is implemented . Specifically, the CNN consists of 6 groups of convolutional, ReLU, and batch normalization, max pooling, and dropout layers, followed by 3 dense layers with ReLU, batch normalization, and dropout. The CNN’s test accuracy on raw testing samples is 98.46%. It consists of more than 59 million parameters and requires 226 MB memory space. Thus, this CNN is heavy for smartphones. Continuously running it on smartphone drains battery quickly. Thus, running it in the cloud is preferred. The overhead for the phone to transmit the images to the cloud is low. Assuming the phone records an image every five seconds and no image compression is applied, the phone only needs a bandwidth of 92 kbps to sustain the transmission, which is little for today’s broadband cellular connectivity.

The MaskNet uses a two-layer MLP architecture. For both the input and output layers, the number of neurons is 57,600. The middle layer has 120 neurons. The MaskNet has 13.9 million parameters and requires 53.61 MB memory space. Thus, it is 7x smaller than InferNet in terms of memory usage. Moreover, MaskNet’s dense layers require much less compute time than InferNet’s convolutional layers. HyperNet architecture is similar to Fig. 3.5, with minor changes on neuron numbers for MaskNet compatibility.

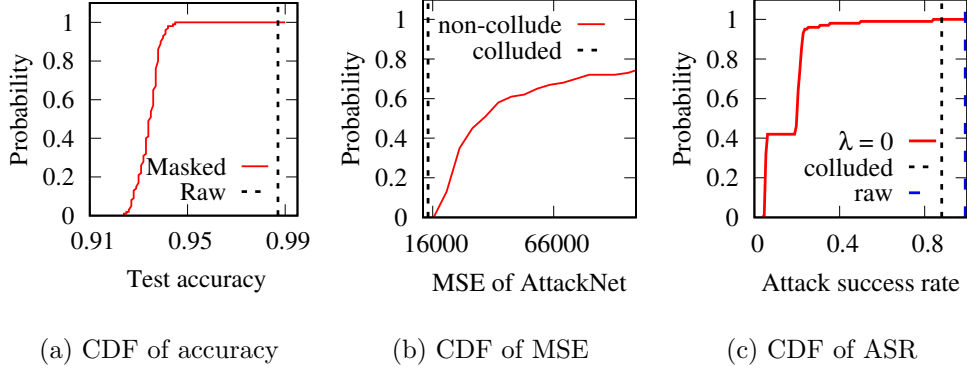


Fig. 3.12: Impact of PriMask on DBR accuracy & privacy.

3.6.2 Evaluation Results for DBR

A HyperNet is trained with $\lambda = 0$. As this HyperNet achieves good privacy preservation as shown shortly and adversarial learning often requires more training epochs, adversarial learning is omitted. Fig. 3.12(a) shows CDF of the test accuracies corresponding to all MaskNets. The average accuracy is 93.4%. Compared with that on the raw data (i.e., 98.46%), there is an accuracy drop of 5.1% on average.

Suppose a mobile colludes with ISP. As shown in Fig. 3.12(b), the ISP achieves an inversion MSE of 14,069. This MSE is much larger than those seen for the MNIST example in §3.3.5, because the MNIST and DBR samples have different pixel value ranges (i.e., $[0, 1]$ vs. $[0, 255]$). Fig. 3.12(b) also shows the CDF of the inversion MSEs for the non-colluding mobiles. Such MSEs are distributed in a wide range from 16,429 to 556,000, with mean and median of 75,484 and 31,304, respectively. Fig. 3.13 shows an original sample in subfigure (a), a non-colluding mobile’s masked data in (b), and ISP’s inversion results for two non-colluding mobiles in (c) and (d). The inversion MSEs of Fig. 3.13(c) and (d) are 28,886 and 16,429, which are smaller than the average MSE and the smallest MSE among the non-colluding mobiles, respectively. Thus, Fig. 3.13(d) is the worst case for the non-colluding mobiles. However, useful information can not be observed, specific to the original sample in Fig. 3.13(a). In this section, only *a priori* information is observed that is applicable to the whole dataset, e.g., rough contours of a car window and a driver. Such *a priori* global information of the DBR application should not be viewed as a particular driver’s privacy. In fact, the ISP can generate an image similar to

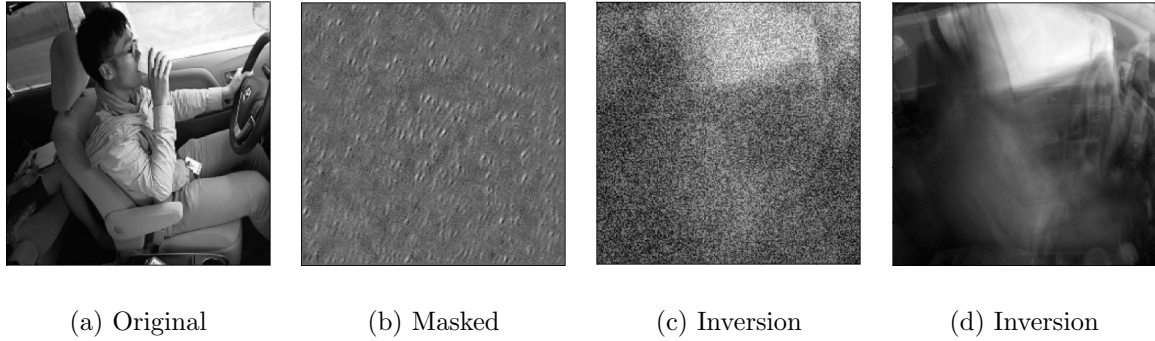


Fig. 3.13: Original, masked, reconstructed samples. The inversion MSE for (d) is the smallest among all non-colluding mobiles, i.e., this example is the worst case for non-colluding mobiles. Only *a priori* global information of the dataset (e.g., rough contours of car window and driver) can be seen from reconstructed samples, which are not specific to a certain driver and thus not private. In fact, ISP can generate an image similar to (d) by averaging all training samples.

Fig. 3.13(d) by averaging all training samples. The example shown in Fig. 3.13 suggests that the HyperNet achieves good privacy preservation against the inversion attack.

PriMask’s resilience against private attribute extraction in the presence of collusion is also evaluated. ExtNet adopts the same architecture as InferNet, except that the last layer has 26 neurons corresponding to the volunteers. On raw data, ExtNet achieves 99.2% ASR. Fig. 3.12(c) shows the ASRs based on raw data and the colluding mobile’s masked data (i.e., 84.4%), as well as CDF of ASRs for non-colluding mobiles. The mean ASR for non-colluding mobiles is 14.8%. Thus, PriMask significantly reduces the attack effectiveness.

3.7 Discussions

Composite privacy threats: In this chapter, the inversion attack and private attribute extraction are mainly handled. The SAL method can be extended to jointly address multiple privacy attacks. Specifically, the composite loss function in Eq. (3.2) can incorporate multiple attack losses (i.e., inversion MSE and ASRs regarding multiple private attributes). During the adversarial learning phase, an InvNet and multiple ExtNets can be trained against a temporary MaskNet. The evaluation of this extended SAL is left to future work.

Validated privacy protection: In §3.4.2.1, the section presented a post-generation validation process to check the quality of a generated MaskNet in terms of InferNet accuracy. Similarly, this chapter can also check in terms of privacy protection against potential mobile-ISP collusion. Now, the private attribution extraction is used as an example to discuss this. Two MaskNets are *conflicting* if the collusion between any of them and the ISP leads to ASR against the other MaskNet higher than a passing threshold. The ASR can be measured using a validation dataset and the two MaskNets' respective ExtNets trained by the PSP. When generating the $(n + 1)^{\text{th}}$ MaskNet, the PSP regenerates the candidate until it does not conflict with any of the previously released n MaskNets. Now, this section analyzes the computation complexity of the above validation process. Assume that the probability that any two freshly generated MaskNets are conflicting is p and the conflict statuses of any two MaskNet pairs are independent. Then, the expected number of generation processes needed for the $(n + 1)^{\text{th}}$ MaskNet is $\frac{1}{(1-p)^n}$. Although the validation process is not scalable in general due to the exponential complexity, it can support a large enough system depending on the needed level of privacy protection. For instance, for the HAR application, the ASR on the raw IMU data is 84%. By setting the ASR passing threshold to 58%, the validation process enables the system with PriMask to provide privacy protection that is validated and better than the system without PriMask. From our measurements, the corresponding conflict probability is about 0.01. As the time for generating an HAR MaskNet is 2.3 ms, if the tolerable validation time is one minute, the validation can support 1,011 mobiles.

Privacy guarantee: The neural network masking approach belongs to a broader category of *instance encoding*. The formal analysis in a recent study [73] has given the theoretical limits of instance encoding in protecting privacy under the notation defined by *distinguishing attack*. However, the privacy guarantee of instance encoding under the notations of inversion attack and private attribute extraction is still an open problem. Despite this uncertainty, instance encoding has been increasingly used in recent approaches for resource-constrained devices [35, 36, 63–67]. This can be due to the practical limitations of other two families of approaches despite their theoretical guarantees [73]: cryptographic techniques (including multiparty computation and homomorphic encryption) incurs large computation and communication overheads; differential privacy is typi-

cally achieved with high utility losses. Nevertheless, the theoretic limits of neural network masking against inversion attack and private attribute extraction deserve future research.

3.8 Summary

This chapter presented PriMask, a cascadable and collusion-resilient data masking approach for mobile devices to use the cloud inference services. In PriMask, the mobile only needs to execute a small-scale neural network called MaskNet to mask the inference data and then sends the result to the cloud. This helps preserve certain private information contained in the inference data. A split adversarial learning method is designed to train a neural network used to generate MaskNet for many mobiles. The heterogeneity of MaskNets provides desirable resilience to the potential collusion between any mobile and the cloud. PriMask is applied to three mobile sensing tasks of human activity recognition, urban environment crowdsensing, and driver behavior recognition. The results show PriMask’s good generalizability and effectiveness in preserving privacy while maintaining the cloud inference accuracy.

Chapter 4

On Lightweight Privacy-Preserving Collaborative Learning for Internet of Things by Independent Random Projections

4.1 Background and Introduction

The recent research advances of machine learning have led to performance breakthroughs of various tasks such as image classification, speech recognition, and language understanding. The drastically increasing amount of data generated by the Internet of Things (IoT) will further foster machine learning performance and enable new applications in various domains.

In particular, *collaborative learning*, which builds a machine learning model (e.g., a supervised classifier) based on the training data contributed by many *participants*, is a desirable and empowering paradigm for smarter IoT systems. By leveraging on the increased volume of training data and coverage of data patterns, collaborative learning will

The work in this chapter has been published as **Linshan Jiang, Rui Tan, Xin Lou and Guosheng Lin. On Lightweight Privacy-Preserving Collaborative Learning for Internet of Things by Independent Random Projections. ACM Transactions on Internet of Things (TIOT). vol. 2, no. 11, pp 1–32. May 2021.** A preliminary version of this work is published by the same authors in **The 4th ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI), April 16-18, 2019, Montreal, Canada.**

approach the intelligence of a crowd and improve the learning performance beyond that achieved by any single participant alone. Moreover, a resource-rich learning *coordinator* (e.g., a desktop-class edge device or a cloud computing service) allows the execution of advanced, compute-intensive machine learning algorithms to capture deeper structures in the aggregated data, whereas the participants (e.g., IoT objects) are often resource-constrained and insufficient for intensive computation. By contributing training data, the individual participants will benefit from the improved machine intelligence in return. However, the data contributed by the participants may contain privacy-sensitive information. Various web services (e.g., webmail and social networking) generally collect and analyze the user data in the raw forms. Thus, users risk their privacy due to both inadvertent or malicious actions by the service provider and due to targeted cyber-attacks by external parties. This risk has been evidenced by several recent large-scale user privacy leak incidents [85–87].

Data anonymization can mitigate the concern; but it is inadequate for privacy preservation, because cross correlations among different databases may be used to re-identify data [88]. Moreover, the correlations between different properties of anonymous individuals (e.g., race, income, political views, etc.) can be exploited to identify people to target for advertisement and advocacy. In the coming era of IoT with many smart objects penetrating into user’s private spaces and times, the current raw data collection approach will only raise significant privacy concerns and may potentially violate relevant laws such as the recent General Data Protection Regulation in European Union and Personal Data Protection Act in Singapore. Therefore, to be successful, IoT-driven collaborative learning applications must preserve privacy.

Privacy-preserving collaborative learning (PPCL) has received increasing research recently under the enterprise settings, where the participants are entities with rich computing resources. The existing approaches can be broadly classified into two categories. The first category of approaches [3, 89–92] follows the distributed machine learning (DML) scheme, such that the participants need not transmit the training data to the coordinator. Instead, the participants and the coordinator will exchange the parameters of machine learning models. The recently proposed *federated learning* [3] is a type of DML. In the second category of approaches [16, 20, 93], each participant applies the homomorphic encryption on the data before being transmitted to the coordinator such that the training

and inference computation can be performed on ciphertexts. However, for resource-constrained IoT objects, these DML and data encryption approaches incur significant and even prohibitive computation overhead. The DML will require the participants to execute machine learning algorithms to train local models, which is often too compute-intensive for IoT objects. Moreover, the iterative communication rounds of DML introduce large communication overhead. Currently, the homomorphic encryption algorithms are still too compute-intensive to be realistic for resource-constrained devices. Therefore, these existing approaches are ill-suited or unpractical for the resource-constrained smart objects beneath at the IoT edge.

This chapter studies the design and implementation of a PPCL approach that is lightweight for resource-constrained participants, while preserving privacy against an honest-but-curious learning coordinator. The coordinator can be a cloud server or a resource-rich fog device, e.g., access points, base stations, network routers, etc. It proposes to apply (1) multiplicative *random projection* at the resource-constrained IoT objects to obfuscate the contributed training data and (2) *deep learning* at the coordinator to address the much increased complexity of the data patterns due to the random projection. Specifically, each participant uses a private, time-invariant but randomly generated matrix to project each plaintext training data vector and transmits the result to the coordinator. This chapter primarily focuses on Gaussian random projection (GRP), because GRP gives several privacy preservation properties of (1) the computational difficulty for the coordinator to reconstruct the plaintext without knowing the Gaussian matrix [94,95], and (2) quantifiable plaintext reconstruction error bounds even if the coordinator obtains the Gaussian matrix [94]. This chapter also considers other random projection matrices such as Rademacher and binary matrices. From a system perspective, random projection is computationally lightweight and does not increase the data volume. Thus, random projection is a practical privacy protection method suitable for resource-constrained IoT objects. Regarding random projection's impact on the design of the machine learning algorithms, the projection can be viewed as a process of mapping the original data vectors to some domain in which the data vectors in different classes are less separable. If the original data vectors are readily separable (that is, they are features), the inverse or pseudoinverse of the random matrix can be considered as a linear feature extraction

matrix. With the deep learning’s unsupervised feature learning capability, this inverse matrix can be implicitly captured by the trained deep model.

To achieve robustness of the privacy preservation against the collusion between any single participant and the curious learning coordinator, each participant should generate its own projection matrix independently. However, this presents a challenge on the PPCL system’s scalability with respect to the number of participants (denoted by N). Specifically, assuming that the training data samples for each class are horizontally distributed among the participants, the number of data patterns for a class will increase from one in the plaintext domain to N in the projection data domain. This increased pattern complexity can be addressed by the strong learning capability of deep learning. Thus, in the proposed PPCL approach, most of the computational workload is offloaded to the resourceful coordinator in the fog or in the cloud. This is different from the existing DML and homomorphic encryption approaches that introduce significant or prohibitive compute overhead to the smart objects beneath at the IoT edge.

To understand the effectiveness of the GRP approach and its scalability with the number of participants, this chapter conducts extensive evaluation to compare GRP with several other lightweight PPCL approaches. The evaluation is based on four example applications with data pattern complexity from low to high. They are handwritten digit recognition, spam e-mail detection, free spoken digital recognition, and vision-based object classification. The baseline approaches include various combinations between (1) multiplicative GRP versus additive noisification for differential privacy (DP) at the participants, and (2) deep neural networks (DNNs), including multilayer perceptron (MLP) and convolutional neural network (CNN), versus support vector machines (SVMs) at the coordinator. The results show that, for the handwritten digit recognition and spam e-mail detection applications with low- and moderate-complexity data patterns, the proposed GRP-DNN approach can support up to hundreds of participants without sacrificing the learning performance much, whereas the GRP-SVM approach may fail to capture the projected data patterns and the performance of the DP-DNN approach is susceptible to additive noisification. The results of this chapter suggest that GRP-DNN is a practical PPCL approach for resource-constrained IoT objects observing data with low- or moderate-complexity patterns. The learning performance and computation overhead of GRP with the Rademacher and binary random projections are also compared.

This chapter implements GRP-DNN, Crowd-ML [89] (a federated learning approach based on shallow learning), and CryptoNets [20] (a homomorphic encryption approach) on a testbed of 14 Raspberry Pi nodes. Experiments show that, compared with GRP-DNN, Crowd-ML incurs 350x compute overhead and 3.5x communication overhead to each Raspberry Pi node. Deep federated learning will only incur more compute overhead. CryptoNets incurs 2.6 million times higher compute overhead to the Raspberry Pi node, compared with GRP.

4.2 Preliminaries

4.2.1 Supervised Collaborative Learning

Supervised machine learning has two phases, i.e., the learning phase and the classification phase. The collaborative learning scheme is formally described as follows. The trained classifier, denoted by $h(\mathbf{x}|\boldsymbol{\theta})$, can classify a d -dimensional data vector $\mathbf{x} \in \mathbb{R}^d$ to be one of a finite number of classes represented by a set \mathcal{C} , where $\boldsymbol{\theta}$ is the classifier parameter and \mathbb{R}^d denotes d -dimensional Euclidean space. The learning process determines the parameter $\boldsymbol{\theta}$ based on the training data. Let N denote the number of participants of the collaborative learning. Let \mathcal{D}_i denote a set of M_i training data samples generated by the participant i , i.e., $\mathcal{D}_i = \{(\mathbf{x}_{i,j}, y_{i,j}) | j \in \{1, \dots, M_i\}, y_{i,j} \in \mathcal{C}\}$, where $\mathbf{x}_{i,j}$ is the training data vector and $y_{i,j}$ is the corresponding class label. For a training data sample consisting of (\mathbf{x}, y) , denote by $l(h(\mathbf{x}|\boldsymbol{\theta}), y)$ the loss function. The collaborative learning solves the following problem to determine the optimal classifier parameter denoted by $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^N \frac{1}{M_i} \sum_{j=1}^{M_i} l(h(\mathbf{x}_{i,j}|\boldsymbol{\theta}), y_{i,j}) + \lambda \|\boldsymbol{\theta}\|^2, \quad (4.1)$$

where the $\lambda \|\boldsymbol{\theta}\|^2$ is the regularization term, $\|\cdot\|$ represents 2-norm, and λ is a parameter affecting the strength of the regularization. With $\boldsymbol{\theta}^*$, the classification for a test data sample \mathbf{x} is to compute $h(\mathbf{x}|\boldsymbol{\theta}^*)$.

A simple approach is to collect all the plaintext training data to the coordinator and solve Eq. (4.1). However, this approach raises the concern of privacy breach, as the raw training data are generally privacy-sensitive. The problem of solving Eq. (4.1) without threatening the participants' privacy contained in \mathcal{D}_i , $i = 1, \dots, N$, is called PPCL.

4.2.2 Random Gaussian Projection (GRP) and Other Random Projections

This section reviews three random projection approaches: GRP, Rademacher random projection, and binary random projection. Note that this chapter primarily focuses on GRP. First, this chapter reviews two properties of GRP. Let $\mathbf{R} \in \mathbb{R}^{k \times d}$ represent a random Gaussian matrix, i.e., each element in \mathbf{R} is drawn independently from the normal distribution $\mathcal{N}(0, \sigma^2)$. GRP has the following two properties [94]:

Property 1 For data vectors $\mathbf{x}_1, \mathbf{x}_2$ and their projections $\mathbf{y}_1 = \frac{1}{\sqrt{k}\sigma} \mathbf{R} \mathbf{x}_1, \mathbf{y}_2 = \frac{1}{\sqrt{k}\sigma} \mathbf{R} \mathbf{x}_2$, the dot product and Euclidean distance between \mathbf{y}_1 and \mathbf{y}_2 are unbiased estimates of those between \mathbf{x}_1 and \mathbf{x}_2 , i.e., $\mathbb{E} [\mathbf{y}_1^\top \mathbf{y}_2] = \mathbf{x}_1^\top \mathbf{x}_2$ and $\mathbb{E} [\|\mathbf{y}_1 - \mathbf{y}_2\|_2^2] = \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2$. The estimation error bounds are $\text{Var}[\mathbf{y}_1^\top \mathbf{y}_2] \leq \frac{2}{k}$ and $\text{Var} [\|\mathbf{y}_1 - \mathbf{y}_2\|_2^2] \leq \frac{32}{k}$.

Property 2 Given a Gaussian matrix instance $\mathbf{R} \in \mathbb{R}^{k \times d}$ where $k < d$ and the projection $\mathbf{y} = \frac{1}{\sqrt{k}\sigma} \mathbf{R} \mathbf{x}$, the minimum norm estimate of \mathbf{x} , denoted by $\hat{\mathbf{x}}$, is an unbiased estimate of \mathbf{x} , i.e., $\mathbb{E} [\hat{\mathbf{x}}] = \mathbf{x}$. The estimation error for the i th element of \mathbf{x} is $\text{Var}[x_i] = \frac{2}{k} x_i^2 + \frac{1}{k} \sum_{j \neq i} x_j^2$.

Based on Property 1, the study [94] shows that a trained SVM classifier can be transferred to classify the projected data. In a recent study [96], a random projection layer that can be implemented by GRP is added to an MLP for dimension reduction. Such design is also based on Property 1. However, the studies [94, 96] do not address collaborative learning and privacy. The estimation error given by Property 2 will be used in the later sections of this chapter to measure the degree of privacy protection provided by the proposed approach.

Rademacher and binary matrices have also been used for random projections [97, 98]. In a Rademacher random matrix, each element is either $\frac{1}{\sqrt{M}}$ or $-\frac{1}{\sqrt{M}}$ with a probability of 0.5, where M is the number of rows in the matrix. In a binary random matrix, each column of the matrix has S ones and $M - S$ zeros, where S is a small integer and M is the number of rows. The position of the S ones are uniformly distributed in a column.

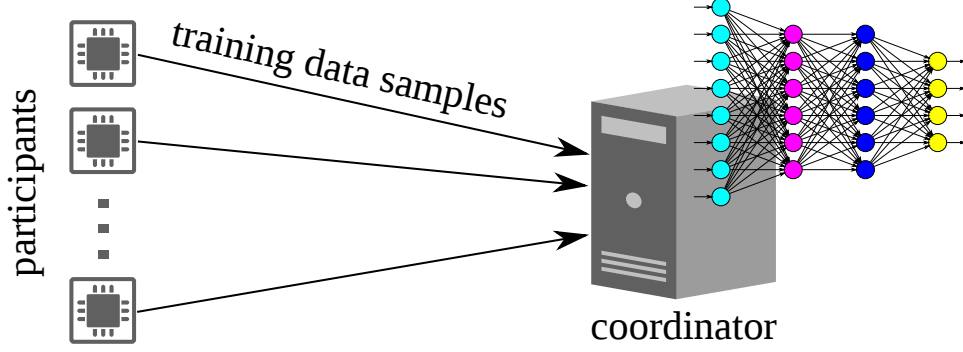


Fig. 4.1: A collaborative learning system.

4.3 Problem Statement and Approach

This section states the PPCL problem in §4.3.1 and presents the proposed independent random projection approach in §4.3.2. §4.3.3 provides two illustrating examples for insights into understanding the effect of GRP on training DNN-based classifiers. §4.3.4 discusses two other alternative approaches for lightweight PPCL and their limitations.

4.3.1 Problem Statement

This chapter considers a PPCL system with N resource-constrained *participants* and an honest-but-curious *coordinator* with sufficient computation power. It is assumed that the data distributed among the participants is homogeneous. Thus, the participants will contribute data in the same format. Fig. 4.1 illustrates the system. During the learning phase, the participants contribute training data samples to build a supervised classifier. As discussed in §4.2.1, the training dataset \mathcal{D}_i contributed by the participant i consists of M_i data vectors $\{\mathbf{x}_{i,j} | j \in \{1, \dots, M_i\}\}$ and the corresponding class labels $\{y_{i,j} | j \in \{1, \dots, M_i\}\}$. As the learning process is often compute-intensive, most of the learning computation should be accomplished by the coordinator. This chapter focuses on addressing the problem of building an effective supervised classifier while protecting certain privacy contained in the data vectors. Several aspects of the problems are now discussed as follows.

The privacy concern regarding the data vectors is primarily due to the fact that the data vectors may contain information beyond the classification objective in question.

For example, consider a PPCL system for training a classifier to recognize human body activity (e.g., sitting, walking, climbing stairs, etc). The recognition is based on various body signals (e.g., motion, heart rate, breath rate, etc) that are captured by wearable sensors. However, the raw body signals can also be used to infer the health statuses of the participants and even pinpoint which people have certain diseases.

This chapter adopts the following threat and privacy models.

Threat model: It consists of the following three aspects:

- *Honest-but-curious coordinator:* This chapter assumes that the coordinator will honestly coordinate the collaborative learning process, aiming to train the best supervised classifier. Thus, it will neither tamper with any data collected from or transmitted to the participants. However, the coordinator is curious about the participants' private information contained in the training data vectors. The coordinator may analyze the data received from the participants to infer the participants' privacy. For instance, the coordinator may attempt to reconstruct and manually inspect the original data captured by the participants.
- *Potential collusion between participants and coordinator:* This chapter assumes that the participants are not trustworthy in that they may collude with the coordinator in finding out other participants' private information contained in the data vectors. The colluding participants are also honest, i.e., they will faithfully contribute their training data to improve the supervised classifier. However, the colluding participants may reveal the details of the adopted privacy-preservation approach to the coordinator. Thus, the design of the PPCL system should maintain the privacy for a participant when any or all of the other participants are colluding with the coordinator.
- *No known input-output attack on non-colluding participants:* This chapter assumes that the coordinator cannot launch the known input-output attack on the non-colluding participants due to the following reasons. In the known input-output attack, the adversary can train an inverse neural network to reconstruct the original data samples. First, the coordinator cannot access the

original data stored at the participants. Second, in the PPCL approach, the communication channel is merely used for uploading obfuscated data samples and their labels. Thus, in the approach, there is no way for the coordinator to obtain the original input of a non-colluding participant. This is different from the approach [13] in which each participant also uses the communication channel to respond to the coordinator's queries by returning obfuscated public data vectors. Without the known input-output attack, it is computationally difficult (practically impossible) for the coordinator to meaningfully estimate the projection matrix and reconstruct the original data vector [94, 95]. Note that, as the participants apply independent Gaussian random projections, the collusion between some participants and the curious coordinator will not enable the known input-output attack on the non-colluding participants.

Privacy model: The raw form of each data vector contains the participant's private information (e.g., health status) and must be protected from snooping by the curious coordinator. The error in estimating the data raw form by the coordinator can be used as a metric to measure the degree of privacy protection. Data form confidentiality is an immediate and basic privacy requirement in many applications.

Four issues that are related to privacy protection and threat model are discussed in what follows.

- *Training data anonymization:* This chapter aims to support anonymization of the training data. That is, the coordinator should not expect to know the participant's identity for any received training data sample. Moreover, the coordinator cannot determine whether any two training data samples are from the same participant. To achieve the above anonymity, the training data samples can be transmitted in separate sessions via an anonymous communication network [99]. Moreover, the transmissions of the data samples from all participants can be interleaved randomly, such that the coordinator cannot associate the data samples from the same participant by their arrival times. Note that the training data anonymization requirement is not mandatory, because the anonymous communication may incur

large overhead for some resource-constrained IoT objects. However, the design of the proposed PPCL approach will not leverage the participants' identities to support data anonymization.

- *Label privacy:* The class labels $\{y_{i,j} | j \in \{1, \dots, M_i\}\}$ may also contain information about the participant. This chapter does not consider label privacy because the participant willingly contributes the labeled data vectors and should have no expectation of privacy regarding labels. In practice, several means can be taken to mitigate the concern of label privacy leak. First, the training data anonymization mitigates the concern during the learning phase. Second, during the classification phase, if the participant has sufficient processing capability to perform the classification computation, the coordinator may send the trained model to the participant for local execution. Existing studies have enabled the execution of deep models on personal and low-end devices [100, 101]. Low-power inference chips (e.g., Google's Edge TPU [102]) will further enhance low-end devices' capabilities in executing classification models. Note that the studies [100, 101] and the inference chips are not to support the much more compute-intensive training.
- *Other privacy models:* Differential privacy [103] aiming at achieving indistinguishability of different data vectors is another widely used quantifiable privacy definition. However, as discussed in §4.3.4 and evaluated in §4.4, the additive noisification implementation of differential privacy is ill-suited for PPCL.
- Tramer et al.'s work [104] focuses on the threat of model extraction and reversal to duplicate the functionality of the model. Differently, this chapter focuses on the threat from the coordinator on the participants' data privacy. In the problem formulation of this chapter, the deep model trained by the coordinator is also available to the coordinator. Tramer et al.'s work is applicable to the external threats that aims at extracting the coordinator's model. Thus, their work is out of the scope of this chapter.

4.3.2 Gaussian Random Projection Approach

Existing DML and homomorphic encryption approaches incur significant computation and communication overhead due to the many computation/communication rounds and data volume swell. The §4.5 provides benchmark results to show this. Thus, these approaches are not promising for resource-constrained participants. This section describes a GRP-based approach that is computationally lightweight and communication efficient for the participants. The overview of the proposed approach is presented as follows.

At the system initialization, each participant i independently generates a random Gaussian matrix $\mathbf{R}_i \in \mathbb{R}^{k \times d}$, where d is the dimension of the data vector. During the learning phase, the participant i keeps \mathbf{R}_i secret and uses it to project all the training data vectors. The participant i transmits the projected training dataset $\mathcal{D}_i = \{\mathbf{R}_i \mathbf{x}_{i,j}, y_{i,j} | j \in \{1, \dots, M_i\}, y_{i,j} \in \mathcal{C}\}$ to the coordinator. After collecting all projected training datasets $\mathcal{D}_i, i = 1, \dots, N$, the coordinator applies deep learning algorithms to train the classifier $h(\cdot | \theta^*)$. During the classification phase, the participant i still uses \mathbf{R}_i to project the test data vector \mathbf{x} and obtains the classification result $h(\mathbf{R}_i \mathbf{x} | \theta^*)$. As discussed in §4.3.1, the classification computation can be carried out at the participant or the coordinator, depending on whether the participant is capable of executing the trained deep model. In the proposed approach, each participant independently generates its random projection matrix to counteract the collusion between participants and coordinator. The two key components of the proposed approach, i.e., GRP and deep learning on projected data, are discussed in the following section.

4.3.2.1 Gaussian random projection

This work mainly considers Gaussian matrices. Specifically, each element of \mathbf{R}_i is sampled independently from the standard normal distribution [105]. The rationale of choosing Gaussian matrices will be explained in §4.3.3.3. This chapter sets the row dimension of \mathbf{R}_i smaller than or equal to its column dimension, i.e., $k \leq d$. Thus, the GRP can also compress the data vector. This chapter defines the compression ratio as $\rho = d/k$. The understanding regarding the admission of compression into the training data projection is as follows. From the compressive sensing theory [106], a sparse signal can be represented by a small number of linear projections of the original signal and

recovered faithfully. Therefore, in the compressively projected data vector, the feature information still exists, provided that the adopted compression ratio is within an analytic bound [106]. In §4.4, the impact of the compression ratio ρ on the learning performance is evaluated.

With GRP, if \mathbf{R}_i is kept confidential to the coordinator, it is computationally difficult (practically impossible) for the coordinator to generate a meaningful reconstruction of the original data vector from the projected data vector [94,95]. Thus, GRP protects the form of the original data. With sufficient pairs of input and output vectors, the coordinator can train a well-designed deep neural network (e.g., the decoder of an autoencoder) to reconstruct the raw forms of original data vectors. However, as discussed in §4.3.1, the coordinator cannot launch the known input-output attack in this chapter’s considered context. In the worst case where the coordinator obtains \mathbf{R}_i , the estimation error given by Property 2 in §4.2.2 can be used as a measure of privacy protection. Random projection has been used as a lightweight approach to protect data form confidentiality in various contexts [107–110].

4.3.2.2 Deep learning on projected data

Feature extraction is a critical step of supervised learning. With the traditional *shallow learning*, the classification system designer needs to handcraft the feature. As an example, in the study [13], the system trains a regress function to recover the Euclidean distance between any two projected samples as the feature. However, the training of the regress function creates a privacy vulnerability. The proposed approach uses deep learning to avoid involving feature engineering that can potentially introduce privacy vulnerabilities. The emerging deep learning method [111] automates the design of feature extraction by *unsupervised feature learning*, which is often based on a neural network consisting of a large number of parameters. Thus, the deep model is often a tandem of the feature extraction stage and the classification stage. For example, a convolutional neural network (CNN) for image classification consists of convolutional layers and dense layers, which are often considered performing the feature extraction and classification, respectively.

The proposed approach utilizes the unsupervised feature learning capability of deep learning to address the data distortion introduced by the GRP. A simple example system

is now used to illustrate this. In this example, in which there is only one participant and the projection matrix \mathbf{R} is a square invertible matrix. Moreover, this example makes the following two assumptions to simplify the discussion. First, it is assumed that a linear transform $\mathbf{\Psi} \in \mathbb{R}^{f \times d}$ gives effective features of the data vectors, where f is the feature dimension. That is, $\mathbf{f} = \mathbf{\Psi}\mathbf{x}$ is an effective representation of the data vector \mathbf{x} for classification. Second, it is assumed that $\mathbf{\Psi}$ can be learned in the form of a neural network by the unsupervised feature learning. The impact of the random projection on the unsupervised feature learning is now discussed. After the projection, the data vector becomes $\mathbf{R}\mathbf{x}$. Moreover, the linear transform $\mathbf{\Psi}\mathbf{R}^{-1}$ will be an effective feature extraction method, since $\mathbf{f} = (\mathbf{\Psi}\mathbf{R}^{-1})(\mathbf{R}\mathbf{x})$. It is reasonable to expect that the unsupervised feature learning can also build a neural network to capture the linear transform $\mathbf{\Psi}\mathbf{R}^{-1}$, similar to the unsupervised feature learning to capture the $\mathbf{\Psi}$ based on the plaintext training data \mathbf{x} . When the projection matrix is non-invertible, its *pseudoinverse* denoted by \mathbf{R}^+ [112] can be used. As the Gaussian random projection matrix is most likely of full rank [113], the linear transform $\mathbf{\Psi}\mathbf{R}^+$ can be regarded as an effective feature extraction. Similarly, it is reasonable to assume that the unsupervised feature learning can capture the linear transform $\mathbf{\Psi}\mathbf{R}^+$ by a neural network. As a result, the deep model trained using the projected data can still classify future projected data vectors. The §4.3.3 will use a numerical example to illustrate this.

The above discussion based on linear features provides a basis for us to understand how the unsupervised feature learning helps address the distortion caused by the GRP. In practice, effective feature extractions are generally non-linear mappings. Neural network-based deep learning has shown strong capability in capturing sophisticated features beyond the above ideal linear features. This chapter will use multiple datasets to investigate the effectiveness of deep learning to address the distortion caused by the GRP.

As discussed earlier, each participant independently generates a Gaussian matrix to counteract the potential collusion between participants and the coordinator. However, this introduces a challenge to deep learning, because the pattern for a class of projected data vectors from N participants will be a composite of N different patterns. Thus, intuitively, a deeper neural network and a larger volume of training data will be needed to well capture the data patterns with increased complexity due to the participants'

independence in generating their projection matrices. The participants' independence can also cause the following possible situation leads to classification errors: $\mathbf{R}_u \mathbf{x}_u = \mathbf{R}_v \mathbf{x}_v$, where \mathbf{x}_u and \mathbf{x}_v are respectively generated by participants u and v and belong to different classes. However, the probability of the above situation is low, especially when the data vectors are of high dimension. Instead, the overlaps between the distributions of any two classes' projected data vectors should receive attention. Fortunately, advanced machine learning algorithms such as SVM and deep learning can learn the mapping from the space of the input data in which the classes overlap to a different space possibly with higher dimensions in which the classes are separated. This issue will be discussed in detail with examples. Nevertheless, the more complex data patterns due to the independent projection matrix generation do cause a challenge. This chapter conducts extensive experiments to assess how well deep learning can scale with the number of participants, compared with the traditional learning approaches.

4.3.3 Illustrating Examples

This section presents a number of examples to illustrate the intuitions discussed in §4.3.2.

4.3.3.1 A 2-dimensional example

Consider a PPCL system with four participants (i.e., $N = 4$) to build a two-class classifier. The original data vectors in the two classes follow two 2-dimensional Gaussian distributions with means of $[-2, -2]^\top$ and $[2, 2]^\top$, and the same covariance matrix of $[1, 0; 0, 1]$. Fig. 4.2(a) shows the plaintext data vectors generated by the four participants. From the figure, the plaintext data vectors of the two classes can be easily separated using a simple hyperplane. Each participant independently generates a Gaussian random matrix. Figs. 4.2(b)-4.2(e) show the projected data vectors of each participant. It can be seen that the patterns of the projected data vectors are different across the participants. Fig. 4.2(f) shows the mixed projected data vectors received from all participants. Compared with Fig. 4.2(a), the pattern of the mixed projected data from all participants is highly complex. Moreover, no simple hyperplane can well divide the two classes. Two other sets of the random projection matrices for all participants are also generated.

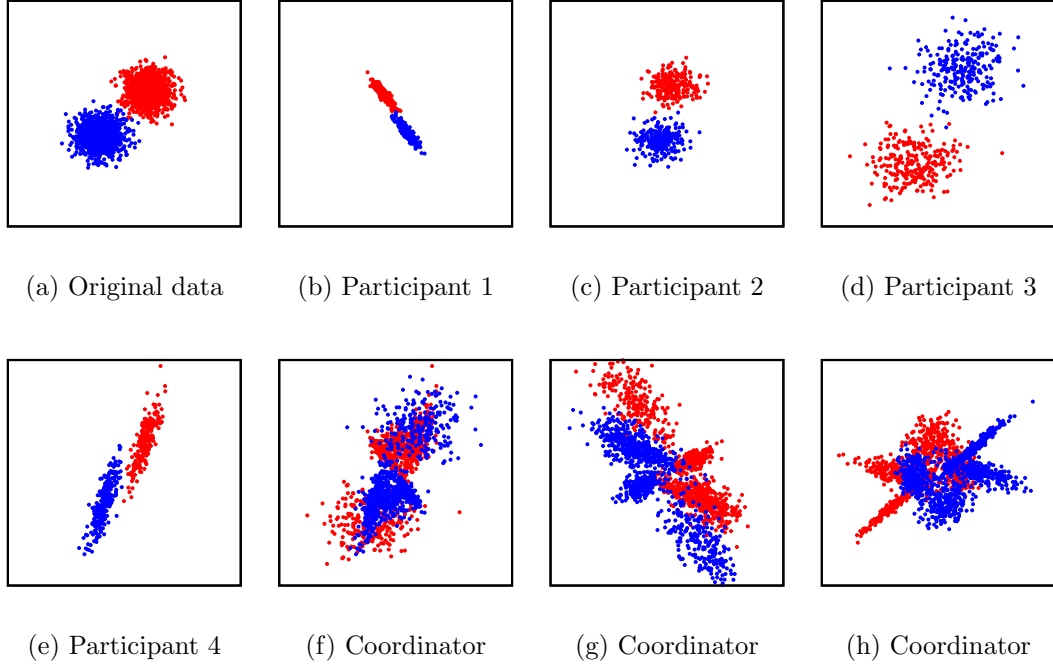


Fig. 4.2: Two-dimensional example. Original data vectors and projected data vectors (red: class 0; blue: class 1). The ranges for the x and y axes are $[-10, 10]$.

Figs. 4.2(g) and 4.2(h) show the mixes of all participants' projected data vectors with the two sets of random projection matrices, respectively. Similarly, the pattern of the mixed projected data from all participants is highly complex.

Then, a classifier based on an MLP with two hidden layers of 30 and 40 rectified linear units (ReLUs) is constructed, respectively. The input layer admits a 2-dimensional data vector, whereas the output layer consists of two ReLUs. The final classification result is generated using a softmax function based on the output layer's ReLU values. Moreover, an SVM classifier as a baseline approach is constructed. LIBSVM [114] is used to implement the classifier. The SVM classifier uses the radial basis function (RBF) kernel with two configurable parameters C and λ . During the training phase, grid search is applied to determine the optimal settings for C and λ .

First, disjoint subsets of the original data is used as shown in Fig. 4.2(a) to train and test the MLP and SVM classifiers. Both classifiers can achieve 99% test accuracy. This shows that the MLP and the SVM are properly designed for the 2-dimensional data vectors.

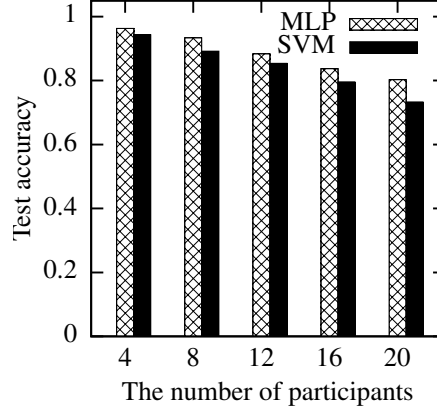


Fig. 4.3: Test accuracy based on projected data vs. the number of participants.

Then, disjoint subsets of the randomly projected data shown in Fig. 4.2(f) is used to train and test the MLP and SVM classifiers. Moreover, the number of participants in the PPCL system is increased. Fig. 4.3 shows the test accuracy versus the number of participants. It can be seen that the MLP classifier always outperforms the SVM classifier. Moreover, the test accuracy decreases with the number of participants. This is because, with more participants, the pattern of the projected data becomes more complex, introducing challenges to both MLP and SVM. The mean test accuracy difference between MLP and SVM increases from 2% to 7%, when the number of participants increases from 4 to 20. This result is also consistent with the understanding that deep learning is more effective in capturing complex patterns than traditional learning.

4.3.3.2 Impact of inter-class overlaps on learning performance

After the participants apply independent GRPs, the consolidated training samples at the coordinator may have inter-class overlaps. A set of numerical experiments based on the previous 2-class 2-dimensional example system is constructed to investigate the impact of the inter-class overlaps on the learning performance.

For each set of the random projection matrices, the cumulative distribution function (CDF) of the Euclidean distance between any two projected data vectors is computed respectively from the two classes. The solid curves in Fig. 4.4(a) are the CDFs, each corresponding to one set of the random projection matrices among the 20 sets. The dashed curve shows the CDF of the Euclidean distance between any two original data

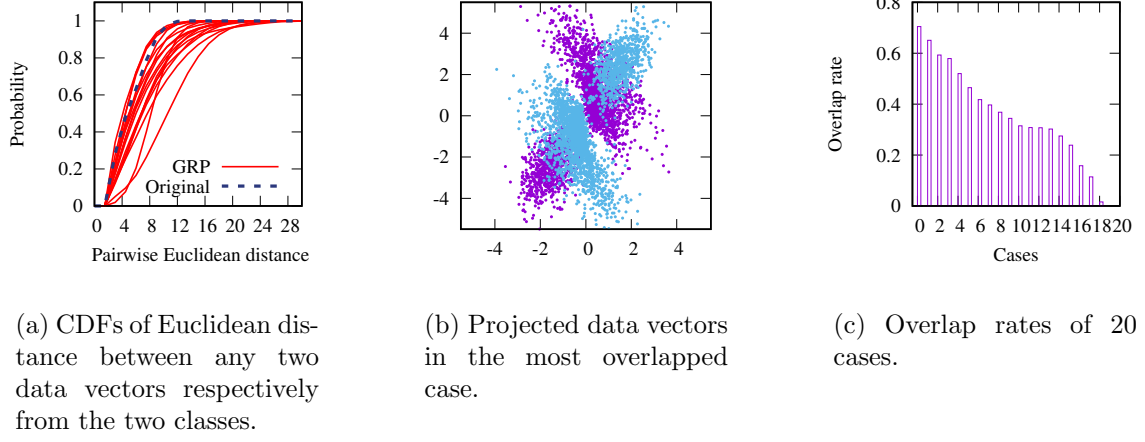


Fig. 4.4: Impact of inter-class overlaps.

vectors respectively from the two classes. It can be seen that the solid curves are in general below the dashed curve, which suggests that the GRPs likely disperse the two classes in terms of inter-sample Euclidean distance.

Fig. 4.4(b) shows the consolidated data vectors after GRPs corresponding to the highest solid CDF curve shown in Fig. 4.4(a). Among the 20 cases, the two classes in the case shown in Fig. 4.4(b) are most overlapped. The inter-class overlap using a metric called *overlap rate* is quantified. It is defined as the ratio of overlapped data vectors to all data vectors. A data vector is overlapped if there are k data vectors of different classes within a distance of r from the considered data vector. In this set of experiments, $k = 3$, $r = 0.01$. Note that as the data vectors shown in Fig. 4.4(b) are distributed in a 10×10 area, the distance threshold $r = 0.01$ is a stringent requirement on the proximity of data vectors in defining overlap. Fig. 4.4(c) shows the ordered overlap rates of the projected data in the 20 cases. The case shown in Fig. 4.4(b) has the largest overlap rate, i.e., 0.705. For this most overlapped case, SVM and MLP achieve test accuracies of 87.24% and 91.08%, respectively, which are still satisfactory. SVM projects the overlapped distributions of the classes to a space with a higher dimension, such that the higher-dimension data distributions of different classes can be separated by linear planes. Compared with SVM, MLP can better handle the overlaps among the data distributions of different classes. The above results show that, although different classes

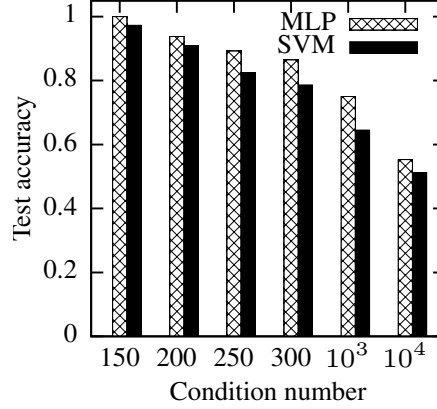


Fig. 4.5: Test accuracy based on projected data vs. the condition number.

may have overlapped areas in the projected data domain, advanced machine learning algorithms such as SVM and MLP may still be able to differentiate the two classes.

4.3.3.3 A 10-dimensional example

Now, another example system is shown to understand the effect of deep learning's unsupervised feature learning capability in addressing the data distortion caused by the random projection. This example is a PPCL system with only one participant (i.e., $N = 1$). The original data vectors in two classes follow two 10-dimensional Gaussian distributions, with the $[-2, -2, \dots, -2]^\top$ and $[2, 2, \dots, 2]^\top$ as the respective mean vectors, and the 10-dimensional identity matrix as their identical covariance matrix.

In the discussions in §4.3.2.2, it is assumed that the projection matrix \mathbf{R} is invertible and the unsupervised feature learning tend to capture $\Psi\mathbf{R}^{-1}$. As learning algorithms are based on numerical computation on the training data, an ill-conditioned matrix \mathbf{R} will impede efficient fitting of $\Psi\mathbf{R}^{-1}$. The subsection verifies this intuition by assessing the learning performance of the single-participant PPCL system using different \mathbf{R} matrices with varying condition numbers. Specifically, by following a method described in [115], the participant generates a random square matrix \mathbf{R} that has a certain condition number value. The condition number is defined as $\|\mathbf{R}\|_F \|\mathbf{R}^+\|_F$ [113], where \mathbf{R}^+ denotes the pseudoinverse of \mathbf{R} and $\|\cdot\|_F$ represents the Frobenius norm. Fig. 4.5 shows the test accuracy of the MLP and SVM classifiers trained using data projected by \mathbf{R} versus the condition number of \mathbf{R} . Note that a larger condition number means that the matrix is

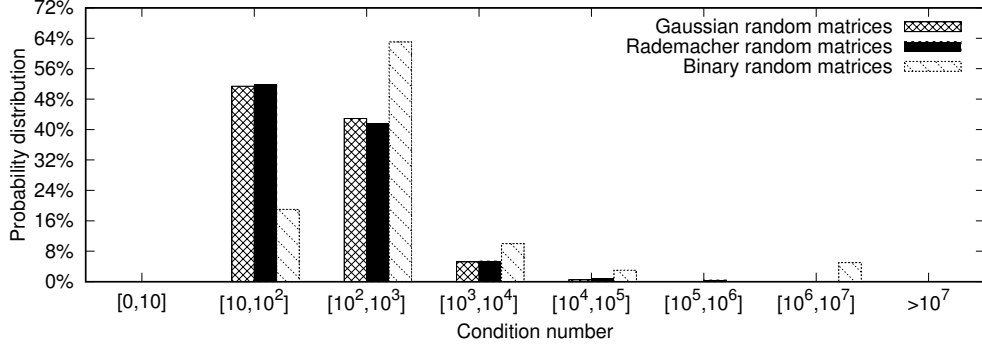


Fig. 4.6: The distributions of the condition number of Gaussian, Rademacher, binary random matrices with dimension 28×28 .

more ill-conditioned. The test accuracy decreases with the condition number, consistent with the intuition.

4.3.3.4 Condition numbers of various projection matrices

§4.3.3.3 shows that the condition number affects the impact of the random projection on the learning performance. This subsection compares the condition numbers of Gaussian, Rademacher, and binary random matrices. The comparison will help understand the superior learning performance of the GRP-based approach. In this section, the Gaussian, Rademacher, and binary random matrices have an identical dimension of 28×28 . For each type of random matrix, 1,000 instances are generated and the distribution of their condition numbers is investigated. Fig. 4.6 shows the distributions of the condition numbers for the three types of random matrices. The condition number distributions of Gaussian and Rademacher matrices are similar, while Rademacher's distribution has a longer tail. Specifically, the probability that a Rademacher matrix's condition number is within $[10^4, 10^5]$ is 0.8%. In contrast, the corresponding probability of Gaussian matrix's condition number within same range is 0.5%. In addition, a binary random matrix can be extremely ill-conditioned. For instance, as shown in Fig. 4.6, the condition number of a binary random matrix can be up to 10^7 . The study [116] has analyzed the distribution of the condition numbers of Gaussian random matrices. The results show that a Gaussian random matrix is well-conditioned with a high probability. For instance, it is shown in [116] that for a 10×5 Gaussian random matrix, the probability that its condition

number is larger than 100 is less than 6×10^{-7} . From the above discussions, Gaussian random matrices are preferred based on their condition numbers. However, Gaussian random projection has higher computation overhead than binary and Rademacher random projections. With a binary matrix defined in §4.2.2, the projection can be implemented using $SN - M$ addition operations. With a Rademacher matrix defined in §4.2.2, the projection can be implemented with $M(N - 1)$ addition operations and just one multiplication operation. In contrast, GRP needs $M(N - 1)$ additions and MN^2 multiplications. Thus, there is a trade-off between the condition of the chosen random matrix type and the associated computation overhead that will be borne by the collaborative learning participants.

4.3.4 Alternative Approaches and Limitations

This section discusses two alternative approaches to PPCL and their limitations. These two alternatives will be used as the baseline approaches in the comparative performance evaluation in §4.4.

4.3.4.1 Non-collaborative learning

If the data anonymity requirement is not enforced, the coordinator can train a separate deep model based on the projected data vectors contributed by each participant. This alternative approach can address the challenge of the complex mixed patterns due to different random projection matrices adopted by different participants as illustrated in §4.3.3. However, it loses the advantages of collaborative learning, i.e., the increased data volume and pattern coverage. From the evaluation in §4.4, compared with the proposed approach, despite that this non-collaborative learning approach additionally uses the participant identity information, it yields inferior average accuracy.

4.3.4.2 Differential privacy

Differential privacy (DP) [103] is a rigorous information-theoretic approach to prevent leak of individual records by statistical queries on a database of these records. The ϵ -DP [103] is formally defined as follows:

Definition 1 A randomized algorithm $\mathcal{A} : \mathbb{D} \rightarrow \mathbb{R}^t$ gives ϵ -DP if for all adjacent datasets $D_1 \in \mathbb{D}$ and $D_2 \in \mathbb{D}$ differing on at most one element, and all $S \subseteq \text{Range}(\mathcal{A})$, $\Pr(\mathcal{A}(D_1) \in S) \leq \exp(\epsilon) \cdot \Pr(\mathcal{A}(D_2) \in S)$.

The ϵ , a positive real number, is a measure of privacy loss, i.e., a smaller ϵ implies better privacy. When ϵ is very small, $\Pr(\mathcal{A}(D_1) \in S) \simeq \Pr(\mathcal{A}(D_2) \in S)$ for all $S \subseteq \text{Range}(\mathcal{A})$, which means that the query results $\mathcal{A}(D_1)$ and $\mathcal{A}(D_2)$ are almost indistinguishable based on any “test criterion” of S . The indistinguishability between the query results $\mathcal{A}(D_1)$ and $\mathcal{A}(D_2)$ decreases with ϵ . The study [117] develops the *Laplace mechanism* of adding Laplacian noises to implement ϵ -DP. Specifically, for all function $\mathcal{F} : \mathcal{D} \rightarrow \mathbb{R}^t$, the randomized algorithm $\mathcal{A}(D) = \mathcal{F}(D) + [n_1, n_2, \dots, n_t]^\top$ gives ϵ -DP, where each n_i is drawn independently from a Laplace distribution $\text{Lap}(S(\mathcal{F})/\epsilon)$ and $S(\mathcal{F})$ denotes the global sensitivity of \mathcal{F} . Note that $\text{Lap}(\lambda)$ denotes a zero-mean Laplace distribution with a probability density function of $f(x|\lambda) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}}$; the global sensitivity is

$$S(\mathcal{F}) = \max_{\forall D' \in \mathbb{D}, \forall D'' \in \mathbb{D}} \|\mathcal{F}(D') - \mathcal{F}(D'')\|_1.$$

Essentially, ϵ -DP gives quantifiable indistinguishability of the query results based on different datasets. The ϵ -DP framework has been applied in various privacy preservation problems in machine learning. The DML approaches to PPCL [89, 90] add random noises to the parameters exchanged between the participants and the coordinator to achieve ϵ -DP. The original parameters can be viewed as deterministic query results of the training data. Adding random noises to the parameters ensures certain levels of indistinguishability between the noise-added parameters based on different training datasets. The achieved ϵ -DP mitigates the privacy concern that the curious coordinator may use the received parameters to infer the existence of particular data vectors in the training dataset. However, these DML approaches [89, 90] incur significant overhead to resource-constrained participants. For PPCL based on resource-constrained participants, an approach to achieving ϵ -DP is to add a Laplacian noise vector to the original data vector \mathbf{x} and then transmit the noise-added data vector to the coordinator for building the classifier. By doing so, certain levels of indistinguishability between the noise-added data vectors based on different original data vectors are achieved.

The recently proposed local differential privacy (LDP) [118] is an ϵ -DP realization different from the Laplace mechanism. It allows statistical computation while protecting each individual user's privacy. As LDP does not require the global sensitivity, it does not depend on the trust in a central authority, which presents practical advantages. However, as shown in [119], LDP needs greater noise levels than the Laplace mechanism and thus reduces the utility of data. Google has implemented LDP in the RAPPOR project [119]. This chapter applies RAPPOR to achieve LDP.

Additive noisification and multiplicative GRP preserve different forms of privacy. Compared with protecting indistinguishability under the DP framework, it is believed that protecting the confidentiality of the raw data form, which can be achieved by GRP, is a more immediate and basic privacy requirement in many applications. The additive noisification, though achieving ϵ -DP, falls short of protecting the confidentiality of the raw data form. Specifically, under the ϵ -DP framework based on zero-mean Laplacian noises, a noise-added data vector can be considered an unbiased estimate of the original data vector with an estimation variance related to ϵ . Thus, the coordinator always has a meaningful (i.e., unbiased) estimate of the raw data. According to Property 2 in §4.2.2, this only happens to the GRP approach in the worst (and unrealistic) case that the projection matrix is revealed to the coordinator; other than the worst case, the coordinator cannot have a meaningful estimate of the raw data form. In the image classification case studies in §4.4, it shows that when ϵ is small (i.e., good DP), the contents of the noise-added images can still be interpreted. In contrast, the projected images cannot be interpreted visually at all.

Applying ϵ -DP to PPCL with resource-constrained participants also introduces the following two challenges:

- *Non-trivial computation overhead:* From the DP theory, an independent random noise vector should be generated and added to every data vector \mathbf{x} . However, random number generation is often a costly operation due to the use of various mathematical functions. The continuous generation of Laplacian noises will incur non-trivial computation overhead for the resource-constrained participants. Differently, in the proposed approach, the random projection matrix generation is a

one-off overhead. The projection to compute $\mathbf{R}\mathbf{x}$ is a lightweight operation consisting of multiplications and additions only. A previous work [108] has implemented the projection operation on an MSP430-based platform. Moreover, the projection can be sped up if a parallel computing chip (e.g., Google’s Edge TPU [102]) is available. In the RAPPOR implementation of LDP, randomized response [120] needs to generate random numbers continuously. Note that continuous random number generation presents substantial overhead to resource-constrained platforms [54].

- *Learning performance degradation:* As discussed in §4.3.2.2, the projection matrix can be implicitly learned by the deep learning algorithms. Differently, the additive Laplacian noises to ensure ϵ -DP can be considered neither a pattern nor an embedding that can be learned by learning algorithms. Thus, the Laplacian noises will only negatively affect the learning performance. Similarly, the random response mechanism of LDP cannot be considered as a pattern that can be learned. The evaluation in §4.4 shows that both the Laplace mechanism and RAPPOR significantly degrade the learning performance.

From the above discussions and the evaluation results in §4.4, adding Laplacian noises to the training data for ϵ -DP is not a promising approach to PPCL with resource-constrained participants.

4.4 Performance Evaluation

The accuracy achieved by various approaches is extensive compared in this section. The computation and communication overhead of these approaches will be profiled in §4.5 based on their implementations on a testbed. The source code of the evaluation can be found from [121].

4.4.1 Evaluation Methodology and Datasets

Extensive evaluation to compare several approaches is conducted here:

- **GRP-DNN:** This is the main proposed approach consisting of GRP at the participants and collaborative learning based on a DNN at the coordinator. The design

or choice of the DNN model will be application specific. The DNN models and training algorithms are implemented based on PyTorch [76].

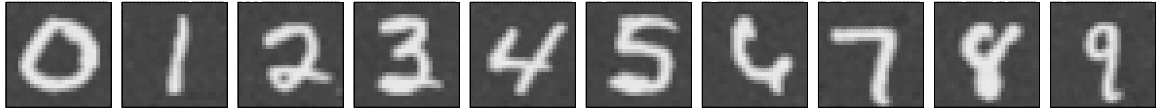
- **RRP-DNN:** This approach replaces the GRP in GRP-DNN with Rademacher random projection (RRP). The DNN models and training algorithms are same as GRP-DNN.
- **BRP-DNN:** This approach replaces the GRP in GRP-DNN with binary random projection (BRP). The DNN models and training algorithms are same as GRP-DNN.
- **GRP-SVM:** This baseline approach applies GRP at the participants and trains an SVM-based classifier at the coordinator. The SVM-based classifier is implemented using LIBSVM [114]. The classifier uses RBF kernel with two configurable parameters C and λ . During the training phase, this chapter applies grid search to determine the best settings for C and λ . This grid search is often lengthy in time (e.g., several days).
- **GRP-NCL:** This is the non-collaborative learning (NCL) baseline approach described in §4.3.4.1. It runs GRP at the participants and trains a separate DNN for each participant at the coordinator. Compared with other approaches, this approach additionally requires the identity of the participant for each training sample.
- **ϵ -DP-DNN:** As described in §4.3.4.2, this approach implements ϵ -DP by adding Laplacian noise vectors to the data vectors and performs collaborative deep learning based on a DNN at the coordinator. Note that this implementation corresponds to the case where $\mathcal{F}(D)$ defined in Definition 4.1 returns D itself. This case is more related to the privacy objective of protecting the raw form of the original data vector. If the DP noises are added to a certain statistics as usually performed in DP applications, the relationship between the additive perturbation and the objective of protecting the raw data form is weakened. As a result, the DP approach and the proposed GRP approach become less comparable. Thus, the DP implementation adds noises to the individual records.



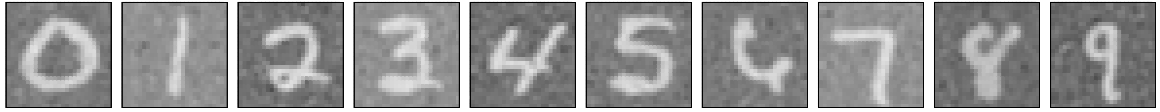
(a) Original images



(b) Projected images in GRP-DNN



(c) Noise-added images in ϵ -DP-DNN ($\epsilon = 50$)



(d) Noise-added images in ϵ -DP-DNN ($\epsilon = 10$)

Fig. 4.7: Example images from MNIST dataset.

- **ϵ -DP-SVM:** This approach implements ϵ -DP by adding Laplacian noise vectors to the data vectors and performs collaborative learning based on SVM at the coordinator.
- **ϵ -LDP-DNN:** This approach implements ϵ -LDP using RAPPOR [120] and performs collaborative deep learning based on a DNN at the coordinator.
- **CNN, SVM, MLP, ResNet-152:** These are the plain learning approaches based on the CNN, SVM, MLP, and ResNet-152 models, respectively. They do not protect any privacy.

The performance evaluation is performed based on four datasets, i.e., MNIST [122], spambase [123], FSD [124], and CIFAR-10 [125].

- **MNIST:** The MNIST dataset consists of 60,000 training samples and 10,000 testing samples. Each sample is a 28×28 grayscale image showing a single, handwritten digit. Fig. 4.7(a) shows an instance of each digit.
- **Spambase:** The spambase dataset consists of 4,601 samples. Each sample consists of (i) a 57-dimensional feature vector that is extracted from an e-mail message and (ii) a class label indicating whether the e-mail message is an unsolicited commercial e-mail. The details of the feature vector can be found in [123]. As the data volume of this spambase dataset is limited, this chapter applies data augmentation to the spambase by adding zero-mean Gaussian noises, resulting in 40,000 training samples and 400 testing samples.
- **FSD:** The free spoken digit (FSD) dataset consists of 2,000 WAV recordings of spoken digits from 0 to 9 in English. The data is randomly splitted into 80% for training, 10% for validation, and 10% for testing. The mel-frequency cepstral coefficients (MFCC) [126] are extracted as the features to represent a segment of audio signal. MFCC can well represent the pertinent aspects of the short-term speech spectrum. As the recordings are of different lengths, this chapter applies constant padding to unify the number of MFCC feature vectors for each recording. As a result, the extracted MFCC feature vectors over time for each recording form a 20×45 matrix.
- **CIFAR-10:** The CIFAR-10 dataset consists of 60,000 32×32 RGB color images in ten classes, in which 50,000 images are for training and 10,000 images are for testing. The 10 classes are airplanes, cars, birds, cats, deers, dogs, frogs, horses, ships, and trucks. Each class has 6,000 images. Fig. 4.17(a) shows an instance of each class.

These four datasets are chosen because the small sizes of the data vectors are commensurate with the limited computing and communication capabilities of IoT end devices.

Training a spam detector based on user-contributed samples (e.g., e-mails) may cause privacy concerns. Thus, the proposed approach is quite appropriate. The choice of the vision-based character recognition and object classification tasks with the MNIST and

CIFAR-10 datasets allows exploiting the learning capabilities of the latest deep models that are often designed for image classification. Moreover, by using images as the data vectors, the effect of the distortion caused by noise adding or random projection can be visualized for intuitive understanding. The CIFAR-10 images have varying backgrounds and object appearances, i.e., complex patterns. Thus, the vision-based object recognition task using CIFAR-10 is more challenging. Although the character and object recognition tasks are not privacy-sensitive, the results based on MNIST and CIFAR-10 will provide understanding on other image classification-based privacy-sensitive applications, such as collaboratively training a mood classifier using the photos in the album of the users' smartphones. The choice of the FSD dataset is to diversify the application scenarios in evaluating the proposed approach. Recently, voice recognition has been integrated into various smart systems such as smartphones and voice assistants found in households and cars. In many scenarios, voice recordings are privacy sensitive. The proposed approach matches the privacy expectations for PPCL applied to voice recognition. In summary, the evaluation datasets cover image, text, and voice modalities, and represent important IoT applications.

For a PPCL system with N participants, by default both the training and testing samples are split evenly into N disjoint sets. Each set is assigned to a participant. Note that in §4.4.2.3, it evaluates the impact of the horizontal distribution of the data on the learning performance, where the training and testing samples are not evenly distributed among the participants. Under GRP-DNN, GRP-SVM, GRP-NCL, RRP-DNN, and BRP-DNN, each participant independently generates its random matrix and uses the matrix to project its plaintext data vectors. The coordinator trains the deep models and SVM based on the projected or noise-added training data vectors from the participants. The trained deep models and SVM are used to classify the projected or noise-added testing data vectors to measure the test accuracy as the evaluation results.

4.4.2 Evaluation Results with MNIST Dataset

A CNN is designed that is used in the GRP-DNN, GRP-NCL, and ϵ -DP-DNN approaches. The CNN consists of two convolutional layers and three dense layers of ReLUs. Max pooling is used after each convolutional layer to reduce the dimension of data after

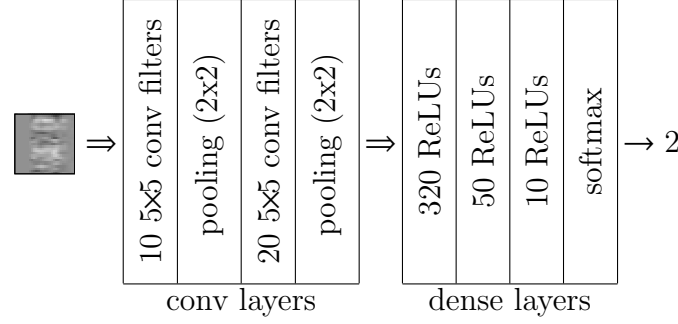


Fig. 4.8: CNN with a projected MNIST image as input.

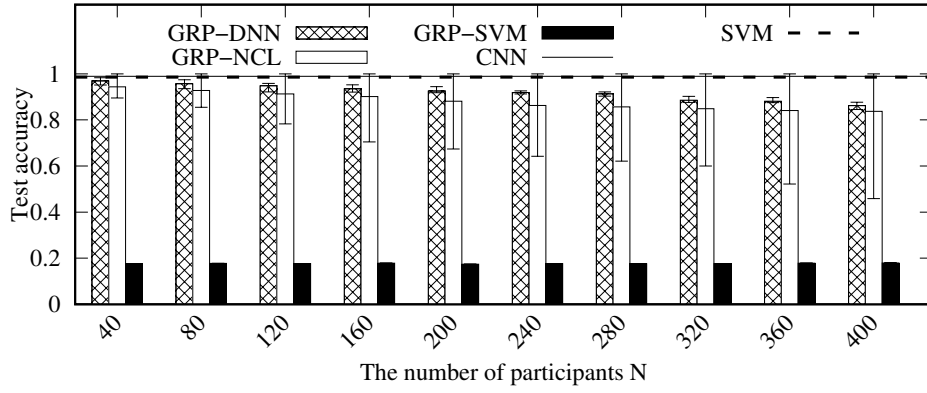


Fig. 4.9: Impact of the number of participants (MNIST). The error bars represent min and max.

convolution. The max pooling controls overfitting effectively and improves the CNN's robustness to small spatial distortions in the input image. The last dense layer has ten ReLUs corresponding to the ten classes of MNIST. A softmax function is used to make the classification decision based on the outputs of the last dense layer. Fig. 4.8 illustrates the design of the CNN. Note that, without random projection, the CNN and the SVM with grid search for kernel parameters achieves test accuracy of 98.7% and 98.52%. This shows that the CNN and SVM capture the patterns of MNIST well.

4.4.2.1 Impact of N on learning performance

The impacts of the number of participants N on the learning performance of GRP-DNN, GRP-NCL, and GRP-SVM are evaluated. The training data and testing data are randomly and equally split into N parts and assigned to N participants. The amount of data with a participant decreases with the increase of N since the total amount of

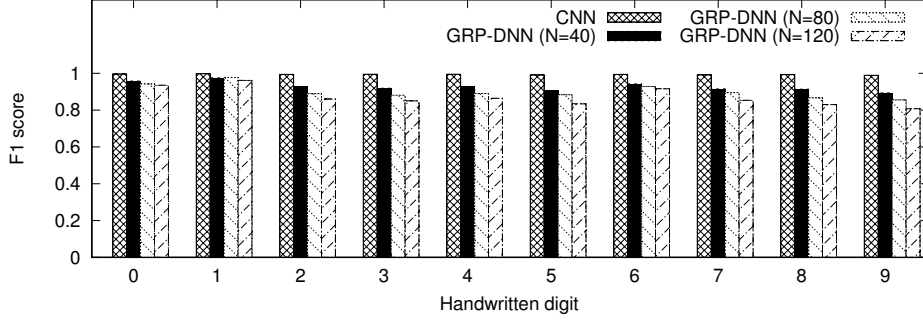


Fig. 4.10: The F1 scores of different handwritten digits in the MNIST dataset under the CNN and the GRP-DNN approaches (MNIST).

data is fixed. Fig. 4.9 shows the results. The two horizontal lines in Fig. 4.9 represent the test accuracy of the plain CNN and SVM without any privacy protection. The two lines overlap. When N increases from 40 to 400, the mean test accuracy of GRP-DNN decreases from 96.87% to 86.18%. If N is no greater than 280, GRP-DNN maintains a test accuracy greater than 90%. The drop of accuracy with increased N is consistent with the understanding that distinct random projection matrices increase the pattern complexity of the aggregated data. However, for MNIST data with light pattern complexities, the GRP-DNN approach can support up to 280 IoT objects for a satisfactory classification accuracy of 90%. Under the GRP-NCL approach, the deep models corresponding to the participants have different test accuracy values. The histogram and error bars in Fig. 4.9 represent the average, minimum, and maximum of the test accuracy values across all trained deep models. Under each setting of N , the maximum test accuracy is 100%. However, the average test accuracy is consistently lower than that of GRP-DNN. This shows that the GRP-NCL that needs to compromise data anonymity yields inferior average learning performance compared with GRP-DNN. This result shows the advantage of collaborative learning. Lastly, the GRP-SVM approach gives poor test accuracy around 17.5% because no efficient RBF kernels can be found to create proper hyperplanes for classification. This observation suggests that DNNs are more efficient in coping with the distortions caused by projections.

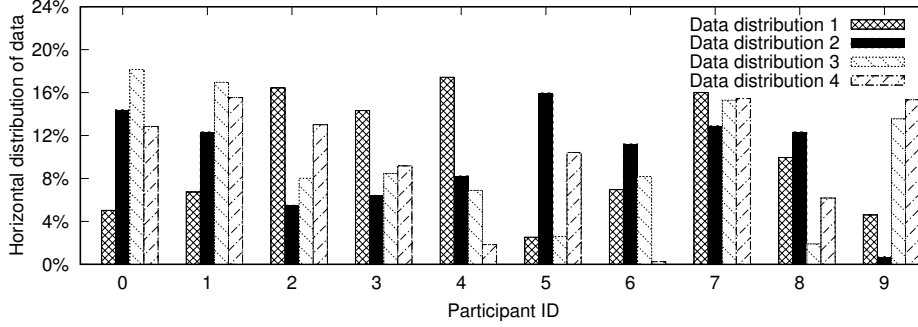


Fig. 4.11: Four horizontal distributions of the training data among 10 participants. The test accuracies for the four distributions are 96.17%, 96.33%, 96.24%, 96.32%, respectively (MNIST).

4.4.2.2 Classification accuracy of different classes

The F1 scores of different classes (i.e., different handwritten digits) under the GRP-DNN and the plain CNN approaches are evaluated. The F1 score of a particular class characterizes the classification accuracy for the class. Thus, from the F1 score distribution among all classes, it can assess whether the classifier is biased for certain classes. Fig. 4.10 shows the results. The F1 score distributions of the GRP-DNN with 40, 80 and 120 participants are similar with the F1 score distribution of the plain CNN. Thus, the DNN trained with the projected data is not biased towards certain classes.

4.4.2.3 Impact of the horizontal distribution of data

In practice, different participants may have different amounts of training data. This set of experiments evaluates the impact of the horizontal distribution of the training data on the learning performance. Fig. 4.11 shows four different horizontal distributions of the training data among 10 participants. During the collaborative learning phase, the participants contribute different amounts of training data. During the classification phase, the horizontal distribution of the testing data is same as that of the learning phase. The corresponding test accuracies of the four horizontal distributions are 96.17%, 96.33%, 96.24%, and 96.32%, respectively. From the results, the horizontal distribution of the data has little impact on the collaborative learning performance.

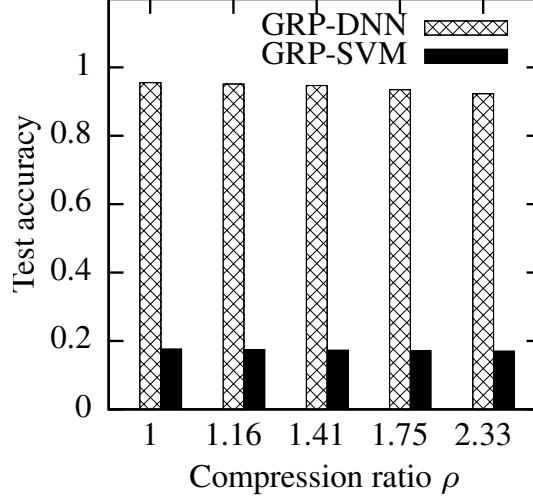


Fig. 4.12: Impact of data compression on learning performance (MNIST, $N = 100$).

4.4.2.4 Impact of data compression

This subsection evaluates the impact of GRP’s data compression on the learning performance. Fig. 4.12 shows the results when $N = 100$. When the compression ratio increases from 1 (i.e., no compression) to 2.33 (i.e., 43% of data volume is retained), the test accuracy of GRP-DNN decreases from 95.52% to 92.85% only. From the discussion in §4.3.2.1, the good tolerance of GRP-DNN against data compression is due to the high sparsity of the MNIST images. In contrast, the GRP-SVM approach performs poorly under all compression ratio settings.

4.4.2.5 Various random projection approaches

This set of experiments compares the performance of collaborative learning from the data obfuscated using GRP, RRP, and BRP. Fig. 4.13 shows the test accuracy of GRP-DNN, RRP-DNN, and BRP-DNN when the number of participants N varies. For all three projection approaches, when N increases from 40 to 400, the test accuracy drops. The GRP-DNN approach gives higher test accuracy than the other two approaches. Recall that §4.3.3.3 has shown the better condition of Gaussian random matrices compared with Rademacher and binary random matrices. The results here are consistent with the understanding that better condition numbers will lead to better learning performance. The learning performance of GRP-DNN, RRP-DNN, BRP-DNN is compared when the

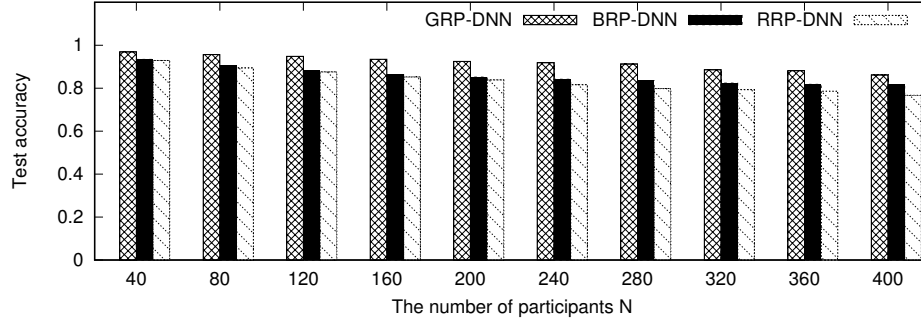


Fig. 4.13: The test accuracy of GRP-DNN, BRP-DNN, RRP-DNN when N varies (MNIST).

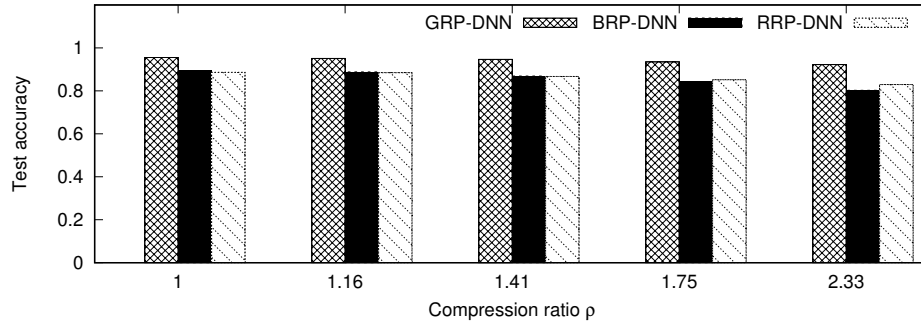
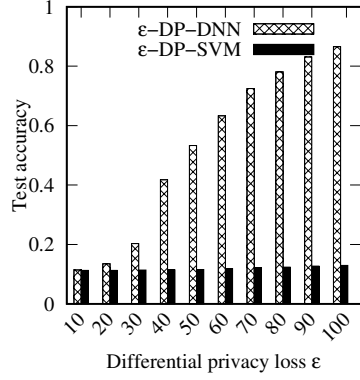
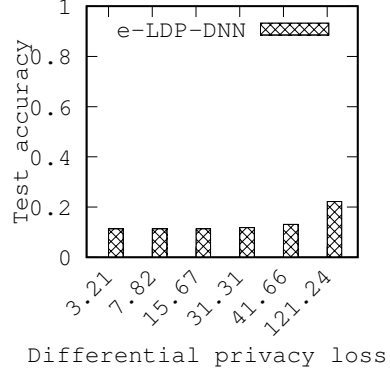


Fig. 4.14: The test accuracy of GRP-DNN, BRP-DNN, RRP-DNN when the compression ratio varies (MNIST, $N = 100$).



(a) Impact of privacy loss of DP on learning performance (MNIST).



(b) Impact of privacy loss of LDP on learning performance (MNIST).

Fig. 4.15: Impact of privacy loss of DP and LDP on learning performance (MNIST).

compression ratio ρ varies. The number of participants is 100. Fig. 4.14 shows the results. When the compression ratio increases, the test accuracy of all the three projection approaches decreases. From Fig. 4.14, in terms of test accuracy, GRP-DNN outperforms RRP-DNN and BRP-DNN.

4.4.2.6 Impact of DP noises

This set of experiments evaluates the impact of adding Laplacian noises to implement ϵ -DP and RAPPOR to implement LDP on the learning performance. Fig. 4.15(a) shows the test accuracy of ϵ -DP-DNN versus the privacy loss level ϵ . Under the considered ϵ -DP-DNN or ϵ -DP-SVM approaches, an ϵ setting smaller than 1 (which is the usual ϵ setting range [103]) will lead to large noise levels such that the learning performance is very poor. To achieve the learning performance comparable to that of the proposed GRP approach, this set of experiments relaxes the range for ϵ . When $\epsilon = 100$ (small Laplacian noises and large differential privacy loss), the ϵ -DP-DNN achieves a test accuracy of 86.6%, lower than those achieved by GRP-DNN when N is up to 400. When $\epsilon = 10$, the performance of ϵ -DP-DNN drops to 11.4%, close to the performance of random guessing. For comparison, the projected and noise-added images with two ϵ settings in Fig. 4.7 are visualized. From Fig. 4.7(b), the projected images can not be visualized. However, from Figs. 4.7(c) and 4.7(d), the noise-added images are easily interpreted

when ϵ is down to 10. Note that in the evaluation, it uses the same CNN model as shown in Fig. 4.8 for the GRP-DNN, GRP-NCL, and ϵ -DP-DNN approaches. This chapter does not spend special efforts to improve the CNN design in favor of any approach; the CNN fed with the original MNIST images achieves satisfactory performance. The poor performance of ϵ -DP-DNN is consistent with the understanding that the performance of deep learning can be susceptible to small perturbations to the data vectors [19]. There are also systematic approaches to generating adversary examples with small differences from the original samples [127, 128]. The adversary examples will be wrongly classified by the deep models. Special care is needed in the deep model design to improve robustness against human-indiscernible perturbations [19]. Significant noises, which are required to achieve good DP protection, are still open challenges to deep learning. Thus, under the ϵ -DP framework, it is challenging to achieve a desirable trade-off between the privacy protection strength and learning performance.

It is discussed in §4.3.4.2 that the additive noisification for ϵ -DP is ineffective in achieving a good trade-off between learning performance and protecting the confidentiality of the raw forms of the training data. The results of GRP-DNN ($N = 1$, $k = d - 1$) and ϵ -DP-DNN are compared. The worst case is considered for GRP-DNN, i.e., the projection matrix \mathbf{R} is revealed to the curious coordinator. From Property 2 in §4.2.2, the minimum norm estimate of the original data vector by the coordinator will have a per-element variance of about 410 for any MNIST image. Under this setting, GRP-DNN achieves a test accuracy of 94.82%. To achieve the same per-element variance of 410, the ϵ value adopted by the ϵ -DP-DNN should be 18.89. Under this ϵ setting, the test accuracy of ϵ -DP-DNN is only 12.86%.

Fig. 4.15(a) also shows the test accuracy of the ϵ -DP-SVM approach. It performs poorly when $\epsilon \leq 100$. This approach achieves good test accuracy only when the added noises are very small under the settings of $\epsilon = 400$ and $\epsilon = 500$.

The BASIC RAPPOR [119] is adopted for ϵ -LDP-DNN on MNIST dataset. BASIC means that each string can be deterministically mapped to a single bit in the bit array. By arranging the pixels of an MNIST sample into a 8-bit array, this experiment adjusts the parameter f, p, q in BASIC RAPPOR to achieve the required privacy loss ϵ . Fig. 4.15(b) shows the test accuracy of ϵ -LDP-DNN versus the privacy loss level ϵ . When $\epsilon = 3.21$,

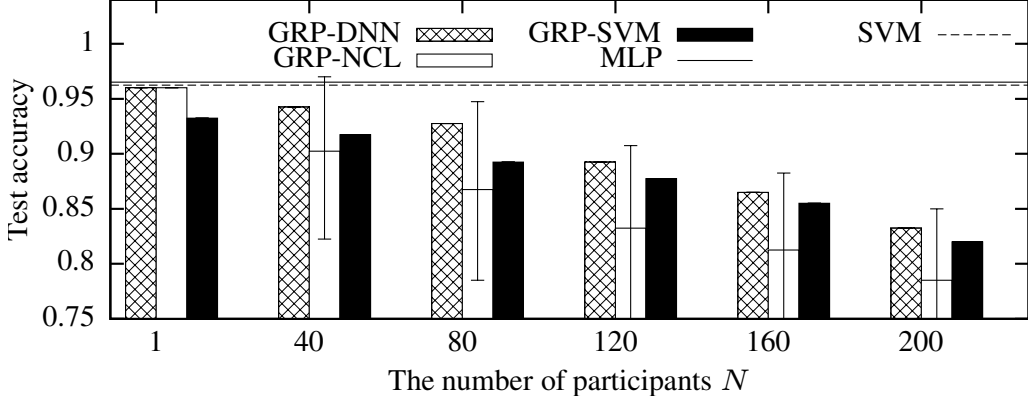


Fig. 4.16: Impact of the number of participants (spambase). The error bars represent min and max.

the ϵ -LDP-DNN only achieves a test accuracy of 11.35%, which is just slightly higher than that of random guessing (i.e., 10%). When $\epsilon = 121.74$, the ϵ -LDP-DNN achieves a test accuracy of 22.21%, much lower than that achieved by ϵ -DP-DNN when $\epsilon = 100$. This result is consistent with the observation in [129] that LDP requires larger noise levels than the Laplace mechanism.

4.4.3 Evaluation Results with Spambase Dataset

This chapter designs a 5-layer MLP classifier to detect spams. The numbers of ReLUs in the five layers are 57, 100, 50, 10, and 2, respectively. A softmax function is used lastly to make the final detection decision. Dropout is used during training to suppress overfitting. Without random projection, the MLP and the SVM with grid research for kernel parameters achieve test accuracy of 96.52% and 96.25%, respectively. This shows that the MLP and SVM can capture the patterns of spambase well.

The impact of the number of participants N on the learning performance of GRP-DNN, GRP-NCL, and GRP-SVM is evaluated here. Fig. 4.16 shows the results. The two horizontal lines in Fig. 4.16 represent the test accuracy of the plain MLP and SVM without any privacy protection. When N increases from 1 to 200, the test accuracy of GRP-DNN decreases from 96% to 83.25%. If N is no greater 100, GRP-DNN can maintain a test accuracy of about 90%. The average test accuracy of GRP-NCL is about 5% lower than that of the GRP-DNN, because GRP-NCL lacks the advantages of collaborative learning. The test accuracy of the GRP-SVM is about 1.25% to 2.75% lower than

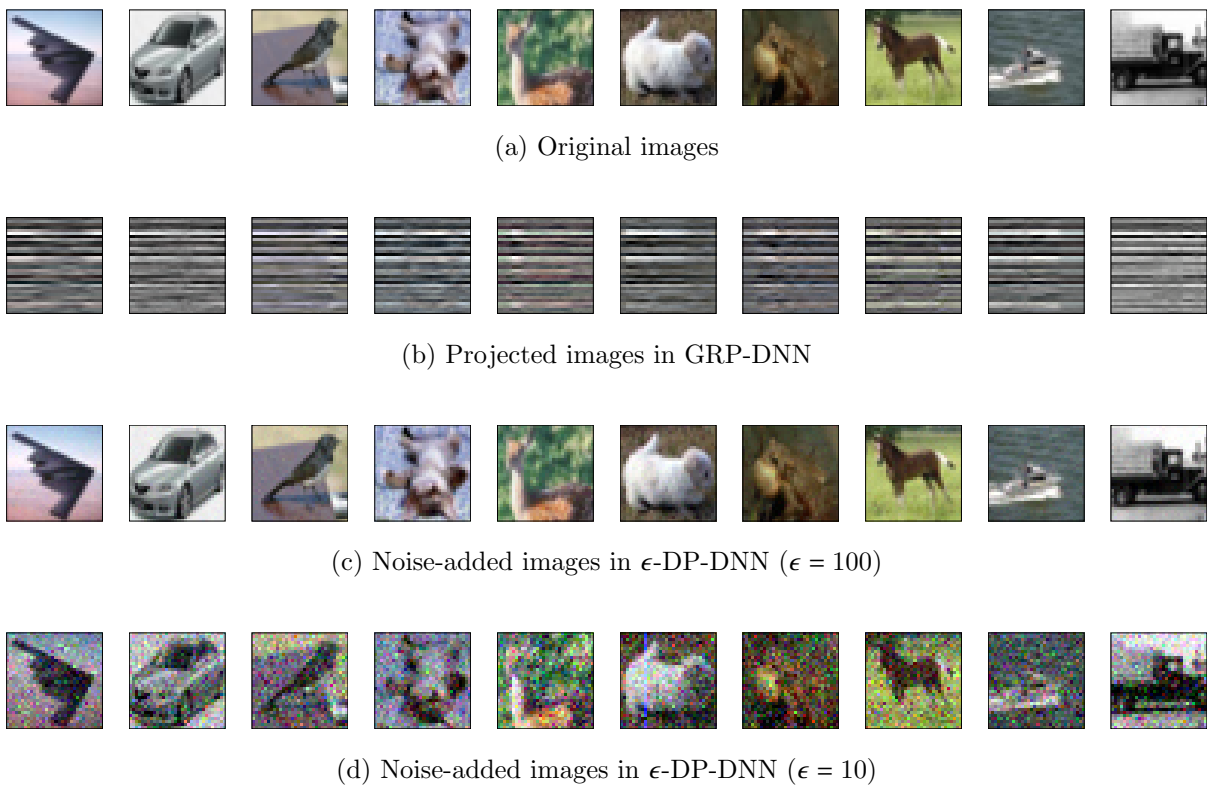


Fig. 4.17: CIFAR-10 image samples. The classes are airplanes, cars, birds, cats, deers, dogs, frogs, horses, ships, and trucks.

that of the GRP-DNN. Thus, the GRP-SVM performs satisfactorily for this spambase dataset. The reasons are two-fold. First, in this spambase dataset, the classifiers operate on the e-mail features, rather than the raw data. Second, the RBF kernel is effective in capturing the features. In fact, the nature of this spambase dataset is similar to that of the 2-dimensional and 10-dimensional generated feature datasets used in §4.3.3, on which the GRP-DNN and GRP-SVM perform similarly.

4.4.4 Evaluation Results with FSD Dataset

This chapter adopts a modified version of the CNN used in [130] to recognize spoken digits. Fig. 4.18 shows the structure of the CNN. The CNN consists of three convolutional layers, one max-pooling layer, and three dense layers. Zero padding is performed to the input image in the convolutional layers and the maxpooling layer. This chapter applies ReLu activation function to the output of every convolutional and dense layer except for the last layer. ReLU rectifies a negative input to zero. The last dense layer has 10 neurons with a softmax activation function corresponding to the 10 classes of FSD. Three dropout layers with dropout rates of 0.25, 0.1 and 0.25 are applied after the max-pooling layer and in the first two dense layers. Specifically, 25%, 10%, and 25% of the neurons will be abandoned randomly from the neural network in the training process. Without random projection, the CNN achieves test accuracy of 98.24%.

The impact of the number of participants N on the learning performance of GRP-DNN in Fig. 4.19 is evaluated. Without any projection, the CNN achieves test accuracy of 98.24%. When N increases from 10 to 50, the test accuracy decreases from 95.27% to 86.21%. The results imply that the proposed approach works well on the FSD Dataset.

4.4.5 Evaluation Results with CIFAR-10 Dataset

To classify the more complex CIFAR-10 images, the residual neural network (ResNet) [131] is adopted. In general, to capture more complex patterns, deeper neural networks will be needed, which often face degraded learning performance, however. ResNet is designed to address this challenge for very deep neural networks. In the experiments, the ResNet-152 is used, which contains 152 layers. Specifically, it consists of *blocks*, each of which consists of convolutional layers and ReLU-based dense layers. After the

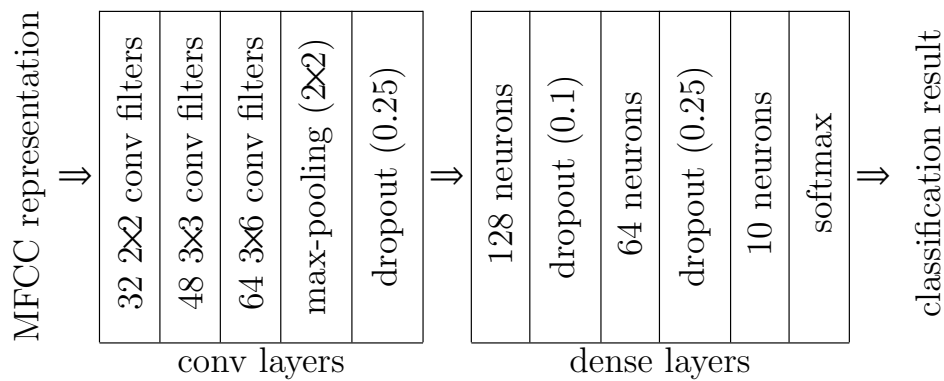


Fig. 4.18: Structure of CNN for FSD recognition.

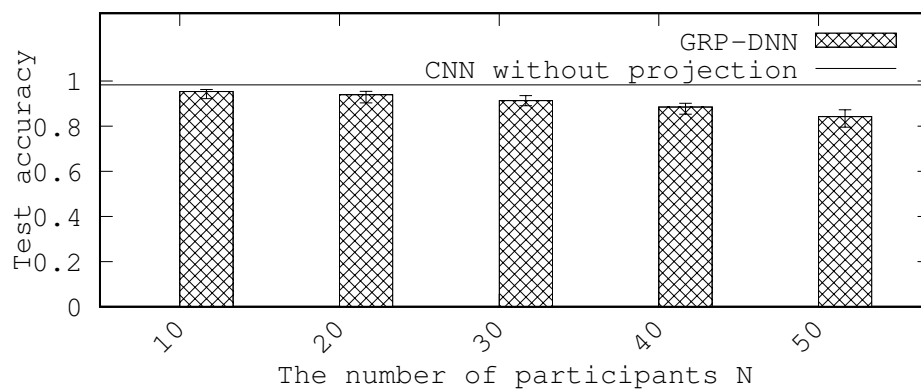


Fig. 4.19: Impact of the number of participants on the learning performance (FSD). The error bars represent min and max.

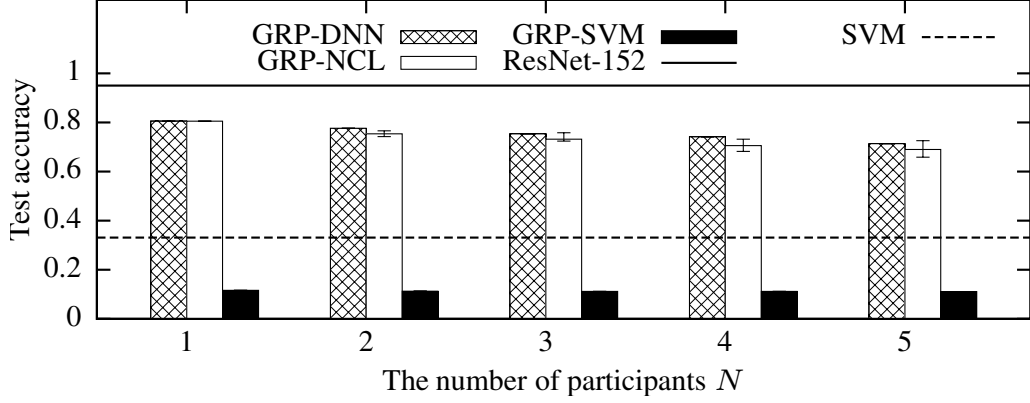


Fig. 4.20: Impact of the number of participants on the learning performance (CIFAR-10).

blocks, ResNet-152 has a fully-connected neural network to make the final classification decision. Without random projection, the ResNet-152 achieves a test accuracy of 95%. This shows that the ResNet-152 can capture the patterns of CIFAR-10 well. In contrast, without random projection, the SVM with grid search for kernel parameters achieves a test accuracy of 33% only. This shows that, due to the high complexity of the patterns in CIFAR-10, no efficient RBF kernels can be found to create proper hyperplanes for classification.

First, the impact of the number of the participants N on the learning performance of different approaches is evaluated. Fig. 4.20 shows the results. The two horizontal lines in Fig. 4.20 represent the test accuracy of the plain ResNet-152 and SVM without any privacy protection. When $N = 1$, the test accuracy of GRP-DNN is 80.6%. Thus, compared with the test accuracy of ResNet-152 without privacy protection, the random projection results in a test accuracy drop of 14.4%. The test accuracy of GRP-DNN decreases with the number of participants. The performance drops are caused by the much more complicated data patterns after the projection, that exceed the complexity that ResNet-152 can handle well. Note that CIFAR-10 had been a challenging dataset until the high accuracy achieved by deep models in recent years. To address the substantially additional pattern complexity introduced by GRP, deeper ResNets may help. But they will require more training data to avoid overfitting. The average test accuracy of the GRP-NCL is slightly lower than the test accuracy of the GRP-DNN. This result is similar to that based on the MNIST and spambase datasets. The test accuracy of GRP-SVM is around 11%, close to that of random guessing.

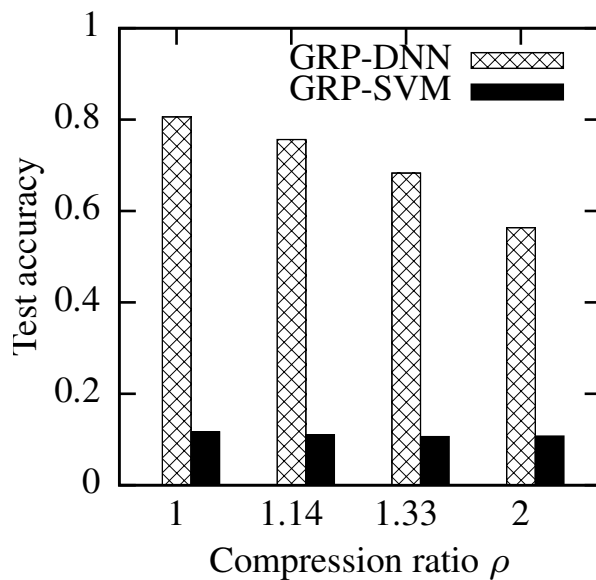


Fig. 4.21: Impact of data compression (CIFAR-10, $N = 1$).

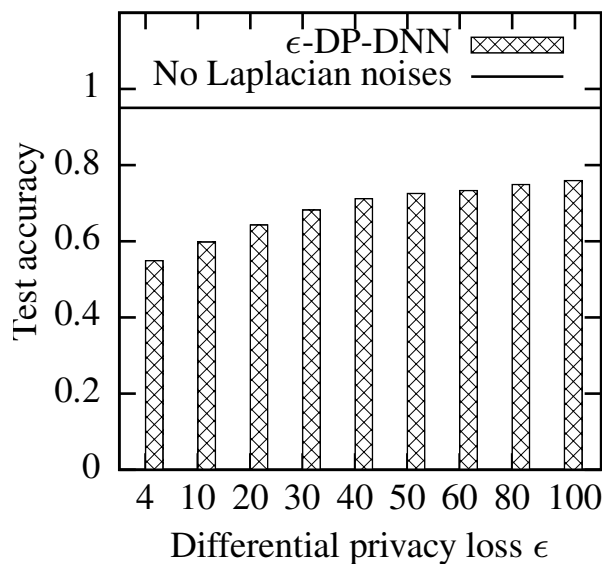


Fig. 4.22: Impact of differential privacy loss (CIFAR-10).

Fig. 4.21 shows the impact of the compression ratio of the projection on the learning performance. The test accuracy of the GRP-DNN decreases with the compression ratio. Compared with the results in Fig. 4.12 for MNIST, the GRP-DNN on the CIFAR-10 is more sensitive to the compression ratio because that CIFAR-10 images are less sparse, and thus less compressible, than the MNIST images. In Fig. 4.21, under all settings for compression ratio, the GRP-SVM's performance is consistently close to random guessing.

Fig. 4.22 shows the test accuracy of ϵ -DP-DNN versus the DP loss level ϵ . When $\epsilon = 100$ (small Laplacian noises and large differential privacy loss), the ϵ -DP-DNN achieves a test accuracy of 75.9%, almost 20% lower than the test accuracy achieved without Laplacian noises. Fig. 4.17(c) shows the noise-added CIFAR-10 images under the setting $\epsilon = 100$. It is almost identical to the original CIFAR-10 images in Fig. 4.17(a). This result echos the understanding that deep learning is not robust to small perturbations [19,127,128]. When $\epsilon = 10$, the content of the noise-added images as shown in Fig. 4.17(d) can still be interpreted. However, from Fig. 4.22, the test accuracy further reduces to 59.8% only. For comparison, Fig. 4.17(b) shows the projected images. The content of the projected images cannot be interpreted.

4.4.6 Summary and Discussion

Several observations from the results in §4.4.2, §4.4.3, and §4.4.5 are as follows.

- Compared with SVM, deep learning can better adapt to the complexity introduced by the multiplicative projections.
- Although the GRP-NCL approach additionally uses the identities of the participants, it gives inferior performance compared with the collaborative GRP-DNN. This shows the advantage of collaborative learning even with the privacy preservation requirement.
- Compared with RRP-DNN and BRP-DNN, GRP-DNN gives higher test accuracy. However, there exists a trade-off between computation overhead and test accuracy in choosing the type of random projection.

- Compared with GRP-DNN, the additive noisification for ϵ -DP achieves inferior trade-off between learning performance and protecting confidentiality of raw forms of training data.
- GRP-DNN shows promising scalability with the number of participants sensing modalities including image, text and voice with low-complexity patterns to be recognized. For the MNIST and spambase datasets, the GRP-DNN can well support 100 participants with a few percents test accuracy drop. For the FSD dataset, the GRP-DNN can support at least 40 participants at a cost of a few percentage points in test accuracy. Besides, as the proposed approach is based on the deep learning in IoT, sufficient amount of labeled training data from each participant is needed. For large-scale PPCL systems involving more participants, a two-tier system architecture can be envisioned as follows. The participants are divided into groups. At the first tier, the proposed GRP-DNN is applied within each group; at the second tier, the DML approach is applied among the group coordinators.

4.5 Implementation and Benchmark

In this section, the overhead of two PPCL approaches are measured (i.e., the GRP-DNN and Crowd-ML [89]) and a privacy-preserving classification outsourcing approach (i.e., CryptoNets [20]) on a testbed of 14 Raspberry Pi 2 Model B nodes [132] and a powerful workstation computer. The Raspberry Pi nodes act as PPCL participants and the workstation acts as the coordinator. They are interconnected using a 24-port network switch. This section benchmarks these approaches using the MNIST dataset. The training and testing samples are evenly allocated to the participants, resulting in 4,285 training samples and 714 testing samples on each participant. The implementations of the three approaches (GRP-DNN, Crowd-ML, CryptoNets) on the same platform, i.e., Raspberry Pi, allow fair comparisons. The participant part of the proposed GRP-DNN can be implemented on mote-class platforms. The previous work [108] has implemented Gaussian matrix generation and GRP on the MSP430-based Kmote platform. However, it is difficult/impossible to implement Crowd-ML and CryptoNets on mote-class platforms.

Table 4.1: The overhead of various approaches.

	Overhead	GRP-DNN	Crowd-ML	CryptoNets
Training	Participant comm. vol.	33.6 MB	117.2 MB	n/a
	Participant compute time	0.96 s	367.24 s	n/a
	Coordinator compute time	928.34 s	1.04 s	n/a
Testing	Participant comm. vol.	5.6 MB	n/a	15.0 MB
	Participant compute time	0.16 s	4.67 s	116 hours
	Coordinator compute time	40.88 s	n/a	

n/a represents “not applicable.”

The GRP-DNN approach is evaluated on the testbed. The compression ratio $\rho = 1$ (i.e., no compression). Table 4.1 shows the benchmark results. During the training phase, each GRP-DNN participant needs to transmit a total of 33.6 MB projected data. A participant can complete projecting all the 4,285 training images within 0.96 s. The coordinator needs 928.34 s to train the CNN. In the GRP-DNN implementation, the testing phase is performed on the coordinator. During the testing phase, each participant completes projecting all the 714 testing images within 0.16 s and transmits a total of 5.6 MB data to the coordinator. The coordinator needs 40.88 s to classify all projected testing images from the participants. Note that GPU acceleration is not used in this benchmark for GRP-DNN during both the training and testing phases.

The Crowd-ML [89] is a DML approach. In Crowd-ML, a participant checks out the global classifier parameters from the coordinator and computes the gradients using its own training data. Then, the participants transmit the gradients to the coordinator that will update the global classifier parameters. Thus, during the training phase, the participants and the coordinator repeatedly exchange parameters. This section applies an existing implementation of Crowd-ML [133] on the testbed. The measurement shows that, during the training phase, each participant needs to upload and download a total of 117.2 MB data, which is 3.5x of the proposed GRP-DNN. The participant compute time is more than 350x of that under GRP-DNN. Despite the larger volume of data exchanges, Crowd-ML achieves 91.28% test accuracy only, which is lower than the 95.58% test accuracy achieved by GRP-DNN. This is because Crowd-ML uses a simple multiclass logistic classifier, which is inferior compared with the CNN used by GRP-DNN in terms of learning performance. Note that during the testing phase of Crowd-ML, the participants

execute their local classifiers. Thus, they do not need to transmit the testing samples to the coordinator for classification.

CryptoNets [20] uses homomorphic encryption to encrypt a testing sample during the classification phase and transmits the encrypted sample to the coordinator. Then, the coordinator uses a neural network trained with plaintext data to classify the encrypted testing sample. Within the homomorphic encryption implementation provided by Microsoft SEAL [134], the homomorphic encryption part of CryptoNets that runs on the Raspberry Pi is implemented. The volume of the 714 encrypted testing images is 15 MB, almost 3x of the data volume generated by random projection. In particular, a Raspberry Pi node takes about 10 minutes and a total of 116 hours to encrypt an image and all the testing images, respectively. This is 2.6 million times slower than the random projection computation. This result clearly shows that the high computation complexity of the homomorphic encryption makes CryptoNets ill-suited for resource-constrained devices.

4.6 Chapter Summary

This chapter proposes a practical privacy-preserving collaborative learning approach, in which the resource-constrained learning participants apply independent random projections on their training data vectors and the coordinator applies deep learning to train a classifier based on the projected data vectors. The proposed approach protects the confidentiality of the raw forms of the training data against the honest-but-curious coordinator. Evaluation using four datasets shows that the proposed approach outperforms various baselines and exhibits promising scalability with respect to the number of participants observing low- to moderate-complexity data patterns. Benchmark on a testbed shows the practicality and efficiency of the proposed approach.

Chapter 5

Differentially Private Collaborative Learning for the IoT Edge

5.1 Background and Introduction

Recent years have witnessed the performance breakthroughs of various pattern recognition tasks due to the research advances in machine learning. In the area of Internet of Things (IoT), many edge devices distributed in urban areas will generate massive data, which can be used to further improve the performance of various machine learning systems. In particular, the *collaborative learning* that builds deep models based on the massive IoT data is envisioned as an important learning paradigm to implement crowd intelligence. In this paradigm, the increased volume and expanded coverage of the training data will significantly improve the quality of the learned model.

However, the training data contributed by the edge devices may contain privacy sensitive information. Data anonymization can mitigate some concern about privacy breach; but it is inadequate, because cross correlations among different databases may be used to re-identify data [88]. Note that recent legislation (e.g., General Data Protection Regulation in European Union and Personal Data Protection Act in Singapore) imposes

The work in this chapter has been published as **Linshan Jiang, Xin Lou, Rui Tan, and Jun Zhao. Differentially Private Collaborative Learning for the IoT Edge. The 2nd International Workshop on Crowd Intelligence for Smart Cities: Technology and Applications (CICS), Beijing, China, 2019, co-located with EWSN.**

stricter requirements for privacy protection. To gain wide adoption in the era of IoT, the collaborative learning systems that rely heavily on the data contributed by the individual edge devices should be designed with proper privacy preservation mechanisms.

This chapter presents the design of a collaborative learning approach that uses the computation capabilities of the edge devices and implements the differential privacy (DP) for the data transmitted to the cloud for building the machine learning model. Specifically, the edge devices collaboratively train a deep neural network, where the training of a number of front layers of the neural network is executed on each edge device and the training of the remaining layers is executed in the cloud. During the training phase, whenever any edge device contributes a training sample, it forward-propagates the training sample over the front layers and transmits the intermediate result data vector to the cloud. The cloud will further forward-propagates the remaining layers to compute the training loss. The training loss is finally fed back to all participating edge devices to update their front layers. Thus, the front layers at all the participating edge devices remain the same during the training phase. On the completion of the training, the layers maintained by the cloud can be disseminated to all the edge devices, such that the whole neural network can be executed locally on the edge devices.

This chapter considers a honest-but-curious cloud that aims to infer private information from the data uploaded by the edge devices during the training phase. The ϵ -DP [103] is adopted as the privacy definition, which gives quantifiable indistinguishability of different data vectors yielded by the edge devices against the honest-but-curious cloud. To implement ϵ -DP, a Laplacian random noise vector is added to the data vector generated by the front layers before being transmitted to the cloud. In the design, batch normalization is applied to the data vector generated by the front layers at the edge device to attain an analytic upper bound of the normalized data. The bound is used as the global sensitivity in setting the Laplacian noise generator parameters to guarantee ϵ -DP.

The proposed approach is applied to a case study of collaboratively training a convolutional neural network (CNN) for image classification. MNIST [122] is used, an image dataset of handwritten digits, to train the CNN. Two convolutional layers with max pooling are trained by the edge devices, while six dense layers are trained by the cloud.

Results show that the proposed approach maintains 99% and 96% classification accuracy in implementing privacy loss levels of $\epsilon = 5$ and $\epsilon = 2$, respectively. Note that, to provide good DP protection, the typical privacy loss level, i.e., ϵ , is often set to a value below 10. For example, in [89], to obtain the balance between system performance and data privacy, the ϵ is set to be 10. In [135], the ϵ is set to 0.5, 2, or 3. Thus, the case study based on MNIST shows that the proposed approach can achieve good DP protection while maintaining satisfactory classification performance.

5.2 Approach Design

This section presents the design of the proposed differentially private collaborative learning approach. The system model and the proposed approach in Section 5.2.1 and Section 5.2.2 are described, respectively. Section 5.2.3 presents an analysis that guides the setting of the Laplacian noise generator to achieve ϵ -DP.

5.2.1 System Model

This chapter considers a collaborative learning system consisting of multiple *learning participants* and an honest-but-curious *learning coordinator* to realize a classification system. In practice, the learning coordinator and participants can be a cloud server and edge devices, respectively. This study only considers the privacy contained in the original data due to potential leakage threat in which the data is used in unauthorized applications. For example, in an activity recognition system based on wearable devices, three-axis acceleration data can be used to infer human body activity. However, the acceleration data can be also exploited to infer the health status of the wearer. With such inferred health status, targeted advertisement can be performed. Thus, protecting the data privacy in a collaborative system is important. Privacy-preserving approach can prevent data abuse.

The coordinator in the system model is honest but curious. Specifically, it honestly supports the collaborative learning process to compute correctly and send results truthfully. However, it is curious about the privacy contained in the data, since it may exploit the privacy for irrelevant applications. This chapter does not consider the privacy contained in the label of the contributed training data since it is assumed that the participant

willingly contributes the labeled data to perform supervised learning and should have no expectation on the privacy contained in the labels. The proposed approach supports anonymization of the training features and labels. Specifically, the coordinator should not expect to know the participant's identity for the received training samples. Moreover, the coordinator cannot determine which two training samples come from the same participant. This can be achieved via an anonymous communication network [99] to transmit the training features and labels to the coordinator.

5.2.2 Approach Overview

This chapter proposes an approach which can protect the privacy of the extracted features before being transmitted to the coordinator to preserve the privacy contained in the original data. Perturbing original data directly to protect privacy may lead to significant learning performance degradation which will be shown in Section 5.3. From the analysis in Section 5.2.3, the results computed from the original data can be perturbed before being transmitted to the coordinator to protect the privacy contained in the original data can be obfuscated.

To realize the advantages of collaborative learning, the classification computation during the learning phase is performed on the coordinator to make good use of various data from different participants. This chapter considers convolutional neural network (CNN) to design collaborative learning system, since CNN is an effective machine learning model. In CNN, convolutional layers fold data in several channels to extract features with specific pooling layers and activation layers. The dense layers (i.e., fully-connected layers) classify the extracted features to yield class labels. In the proposed collaborative learning system, each participant runs convolutional layers to extract features that will be transmitted to the coordinator. The coordinator maintains the dense layers and forward-propagates them with the received features during the learning phase. Moreover, the participants will perturb the features before transmitting them to the coordinator.

Figure 5.1 illustrates the system architecture. Each participant collects data and extracts features locally. Under the privacy-preserving mechanism that will be presented in Section 5.2.3, participant sends privacy-preserving features and original labels to the coordinator such that the coordinator can train the fully-connected layers. The coordinator

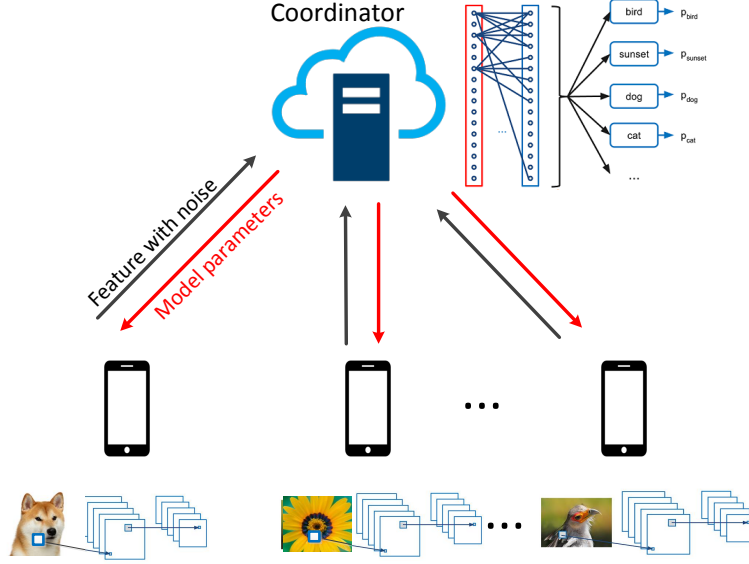


Fig. 5.1: Overview of the proposed privacy-preserving collaborative learning approach.

uses the backpropagation algorithm to update the fully-connected layer parameters and meanwhile sends back the propagated loss to the participants which will update convolutional layers accordingly. Once a participant finishes the above training process among their training data, it sends the parameters of the convolutional layers to the coordinator for updating the convolutional layers in the coordinator. Another participant downloads the updated convolutional layers and repeats the above training process. By repeating the above process for all participants, the convolutional layers of all the participants are updated based on each contributed training data sample. Thus, the participants enjoy the advantages of collaborative learning, which help them better extract features.

Several design issues are discussed as follows.

- During the classification phase after the completion of the collaborative learning, the participant can send testing data features to the cloud, which will then perform classification using the dense layers. Alternatively, on the completion of the collaborative learning, the coordinator can disseminate the dense layers to all participants. Then, each participant can run the full CNN to perform classification without transmitting testing data.

- In order to utilize the large volume of training data to improve the effectiveness of the convolutional layers, it is desirable to maintain the same convolution layers at all the participants. The following method is adopted to keep convolutional layer consistency among participants. After the coordinator updates dense layer parameters, it broadcasts propagated loss to all the participants. Thus, all the participants can update their own convolutional layers simultaneously. Since the same hyperparameters for the convolutional layers at all the participants can be configured, the convolutional layers at all the participants can be maintained consistent.
- The system will have significant overhead if each participant immediately sends new extracted features once it generates new data. To solve this issue, in the proposed design, if the data exceeds a specified value, the participant starts to process data to extract feature and transmit it. This method matches well with the proposed privacy-preserving approach which adopts batch normalization and Laplacian noisification, which will be presented in the next subsection.

5.2.3 Achieving ϵ -Differential Privacy

Differential privacy is an information-theoretic approach to protecting data privacy. It aims to confound the query results based on adjacent datasets. The proposed approach adopts ϵ -differential privacy [103] as the privacy definition. The ϵ -differential privacy (ϵ -DP) is formally defined as follows: *A randomized algorithm $\mathcal{A} : \mathcal{D} \rightarrow \mathbb{R}^t$ gives ϵ -DP if for all adjacent datasets $D_1 \in \mathcal{D}$ and $D_2 \in \mathcal{D}$ differing on at most one element, and all $S \subseteq \text{Range}(\mathcal{A})$, $\Pr(\mathcal{A}(D_1) \in S) \leq \exp(\epsilon) \times \Pr(\mathcal{A}(D_2) \in S)$.* Here, the differential privacy level ϵ , is a positive number which measures privacy loss. Smaller ϵ always means better protection: when ϵ is very small, $\Pr(\mathcal{A}(D_1) \in S) \approx \Pr(\mathcal{A}(D_2) \in S)$ for all $S \subseteq \text{Range}(\mathcal{A})$. Thus, the query results $\mathcal{A}(D_1)$ and $\mathcal{A}(D_2)$ are nearly indistinguishable, which prevents the attackers from recognizing the original dataset. An approach to implementing ϵ -DP is to add Laplacian noise [136]. Concretely, for all function $\mathcal{F} : \mathcal{D} \rightarrow \mathbb{R}^t$, the randomized algorithm $\mathcal{A}(D) = \mathcal{F}(D) + [n_1, n_2, \dots, n_t]^\top$ gives ϵ -DP, where each n_i is drawn independently from a Laplace distribution $\text{Lap}(S(\mathcal{F})/\epsilon)$ and $S(\mathcal{F})$ denotes the global sensitivity of \mathcal{F} . Note that the global sensitivity $S(\mathcal{F})$

is $S(\mathcal{F}) = \max_{\text{neighboring databases } D, D' \in \mathbb{D}} \|\mathcal{F}(D) - \mathcal{F}(D')\|_1$ while $\text{Lap}(\lambda)$ is a zero-mean Laplace distribution with a probability density function of $f(x|\lambda) = \frac{1}{2\lambda} e^{-\frac{|x|}{\lambda}}$.

A challenge in implementing ϵ -DP is the determination of the global sensitivity $S(\mathcal{F})$. It is hard to determine global sensitivity after convolutional layers. Theoretically, the output of convolutional layers can continuously increase or decrease during training epochs. However, too large or too small outputs of the convolutional layers may cause the *gradient exploding problem* or *gradient vanishing problem* [137]. Batch normalization (BN) [137] is developed to normalize the output of hidden layers to avoid the problems to support neural network training. Using each batch as a unit, BN normalizes the output of specific layers and then forwards it to the next layer. It limits the range of the output, enabling the determination of the global sensitivity $S(\mathcal{F})$. The proposed approach applies standard BN parameters: fixed variance 1 and fixed mean 0. The following explains the method to compute the global sensitivity $S(\mathcal{F})$.

For simplicity, it is assumed that there is only one channel in the CNN and the dimension of the output of the convolutional layers is $L \times W$. Denote the batch size in the convolutional neural network as N , the output of convolutional layers in a position $\langle i, j \rangle$ of element k in the batch as $X_{i,j,k}$. The difference between two adjacent datasets D and D' in the current scenario is $X_{i,j,k}$ and $X'_{i,j,k}$, while the other elements are the same. The query request in the current scenario is to read each element in the dataset because the coordinator can access all data sent from the participants. Thus, the global sensitivity $S(\mathcal{F})$ is equal to the maximum difference between $X_{i,j,k}$ and $X'_{i,j,k}$, $S(\mathcal{F}) = \max_{\langle i,j,k \rangle \in \langle L,W,N \rangle} \{X_{i,j,k} - X'_{i,j,k}\}$. Due to the constraint imposed by BN, I have $\sum_{k=1}^N X_{i,j,k} = 0$ and $\sum_{k=1}^N X_{i,j,k}^2 = N$.

To analyze $S(\mathcal{F})$, the following proves that for any $\ell \in \{1, 2, \dots, N\}$ that $-\sqrt{N-1} \leq X_{i,j,\ell} \leq \sqrt{N-1}$ and both equal signs are applicable in special cases.

From the Cauchy–Schwarz inequality, the following can be derived:

$$\left(\sum_{t \in \{1,2,\dots,N\} \setminus \{\ell\}} X_{i,j,t} \right)^2 \leq (N-1) \sum_{t \in \{1,2,\dots,N\} \setminus \{\ell\}} X_{i,j,t}^2. \quad (5.1)$$

Applying $\sum_{t \in \{1,2,\dots,N\} \setminus \{\ell\}} X_{i,j,t} = -X_{i,j,\ell}$ and $\sum_{t \in \{1,2,\dots,N\} \setminus \{\ell\}} X_{i,j,t}^2 = N - X_{i,j,\ell}^2$ to Inequality (5.1), I obtain $X_{i,j,\ell}^2 \leq (N-1) \cdot (N - X_{i,j,\ell}^2)$, which leads to

$$-\sqrt{N-1} \leq X_{i,j,\ell} \leq \sqrt{N-1}. \quad (5.2)$$

The equal sign in the first “ \leq ” of Inequality (5.2) is applicable when $X_{i,j,\ell} = -\sqrt{N-1}$ and $X_{i,j,t} = 1/\sqrt{N-1}$ for $t \in \{1, 2, \dots, N\} \setminus \{\ell\}$. Similarly, the equal sign in the second “ \leq ” of Inequality (5.2) is taken when $X_{i,j,\ell} = \sqrt{N-1}$ and $X_{i,j,t} = -1/\sqrt{N-1}$ for $t \in \{1, 2, \dots, N\} \setminus \{\ell\}$.

From the above analysis, $S(\mathcal{F})$ denoting $\max_{\langle i,j,k \rangle \in \langle L,W,N \rangle} \{X_{i,j,k} - X'_{i,j,k}\}$ is equal to $2\sqrt{N-1}$. By adding a random noise from $\text{Lap}(S(\mathcal{F})/\epsilon)$ [117], the ϵ -DP is achieved to protect original data privacy. Without loss of generality, it also succeeds when there are multiple channels in CNN.

5.3 Performance Evaluation

This section evaluates the proposed approach in an application of image-based handwritten digit recognition.

5.3.1 Evaluation Methodology and Settings

The evaluation is based on a public dataset MNIST [122]. MNIST is a hand written dataset which consists of 60,000 training samples and 10,000 testing samples. Each sample is a 28×28 gray scale image showing a handwritten number within 0 to 9. It is widely used in machine learning literature as a basic benchmark dataset to evaluate the learning performance.

In the considered model, the CNN deployed at the participants has two convolutional layers with 30 and 80 channels, respectively. After each convolutional layer, max-pooling layers is applied to reduce the size of the output. In neural networks, the max-pooling layer can accelerate learning with reduced parameter dimension while extracting features of subregion in a sample. In dense layers, the ReLU activation layer is used to increase the nonlinearity of the neural network. After the second convolutional layer, a BN layer is used to accelerate the learning rate and prevent gradient vanishing problem and gradient exploding problem. Then, the DP technique is applied to perturb the output of the BN layer to preserve data privacy.

After adding the DP noise, the participants send perturbed features to the coordinator as the input for the dense layers. In the proposed model, the dense layers contain four

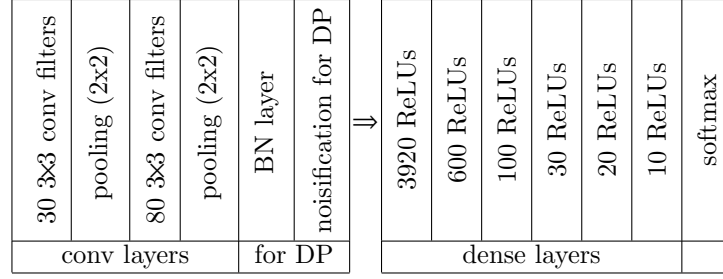
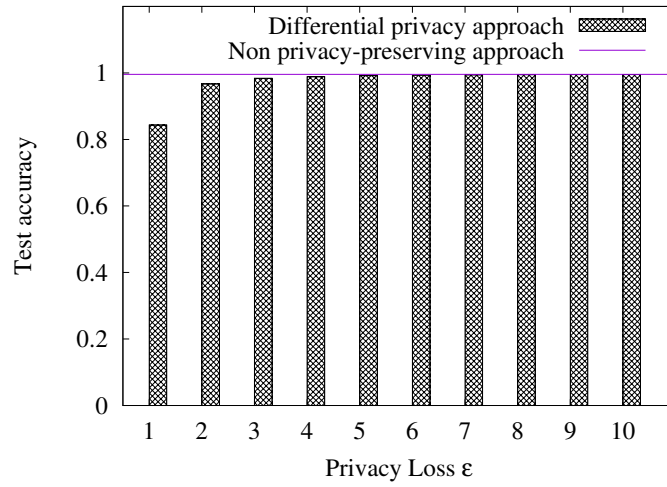


Fig. 5.2: CNN structure.


 Fig. 5.3: Impact of privacy loss level ϵ on the test accuracy of the collaboratively learned model with DP.

hidden layers for reducing the data dimension gradually and one output layer with a dimension of 10 which is the dimension of labels. Finally, softmax layer is used to predict label and compute loss. The structure of the CNN is shown in Figure 5.2.

In the experiments, the hyperparameters of CNN are set as follows: the learning rate is equal to 0.01 and the batch size is equal to 64. Thus, the global sensitivity $\mathcal{S}(\mathcal{F})$ is equal to $\sqrt{63}$. Therefore, various privacy loss levels ϵ are applied to evaluate the performance of the differentially private collaborative learning based on MNIST.

5.3.2 Evaluation Results

For comparison, the centralized training approach without any privacy consideration is used as the baseline. The corresponding CNN excludes the noisification layer as shown

in Figure 5.2. This centralized non-DP approach achieves 99.58% test accuracy. From Figure 5.3, with the proposed differentially private collaborative learning approach, the test accuracy increases with the privacy loss level ϵ . Note that a large ϵ means less privacy protection. Thus, there exists a trade-off between the test accuracy and the degree of privacy protection. Generally, when the ϵ is chosen to be 5, which is often considered providing satisfactory privacy protection [89,135], the proposed approach can still achieve 99.18% test accuracy. When ϵ is reduced to 1, the test accuracy decreases to 84.33%, because large DP noises start to undermine the performance of the classification system. However, when ϵ is around 2 to 5, the system shows good classification performance. Specifically, only 3% of accuracy reduction is observed when ϵ reduces from 5 to 2.

In the second set of experiments, the impact of BN's batch size on the classification performance of the collaboratively learned model is investigated. For training CNN, a smaller batch size often results in more accurate estimation of the gradient descent, but longer convergence time of the training process. Moreover, in the proposed approach, the batch size N determines the global sensitivity $S(\mathcal{F})$, i.e., $S(\mathcal{F}) = \sqrt{N-1}$. Thus, the smaller batch size also results in lower noise levels for the same ϵ setting. The ϵ is set to be 2. Figure 5.4 shows the test accuracy of the CNNs trained by the proposed differentially private collaborative learning approach and the centralized learning approach without privacy preservation, under different batch size settings. When the batch size $N = 32$, the test accuracy is 99.5% and 98.1% for the centralized non-DP learning approach and the proposed DP approach, respectively. When N increases to 128, the accuracy drops to 99.3% for the centralized non-DP approach and 94.0% accuracy for the proposed approach. For the proposed approach, with a larger batch size, both the global sensitivity and noise level become larger, leading to performance drop.

Adding Laplacian noises to the original data to achieve ϵ -DP is an alternative approach. This section investigates its effectiveness. Under this alternative approach, the global sensitivity $S(\mathcal{F})$ of the original data (i.e., the pixel values) is the maximum difference between any two pixels. Since the pixel value in MNIST is within the range of $(0, 255)$, the global sensitivity is a fixed value of 255. Figure 5.5 shows the test accuracy of the CNN trained by this alternative approach under various ϵ settings. It can be seen that, when $\epsilon = 10$, the test accuracy is 11.35% only, which is close to the performance

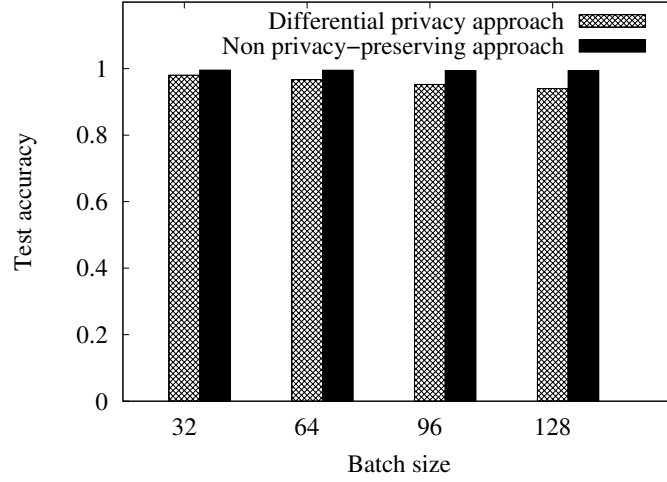


Fig. 5.4: Impact of batch size on the test accuracy of the collaboratively learned model with DP ($\epsilon = 2$).

of random guessing (i.e., 10%). When $\epsilon \geq 100$, although the approach can achieve good test accuracy, the privacy loss is too high to be meaningful in a collaborative learning system. Thus, the results show that adding Laplacian noises to the original data significantly degrades the learning performance. Moreover, by comparing the results obtained with this alternative approach and the proposed approach, it can be seen that the unsupervised feature learning performed by the convolutional layers is susceptible to the DP noises, whereas the classification boundary learning performed by the dense layers is more robust to the DP noises.

5.4 Chapter Summary

This chapter presents the design of a collaborative learning approach that trains different stages of a deep neural network at the edge devices and the cloud, respectively. The deep neural network model is constructed based on the training samples contributed by all the participating edge devices. To protect the privacy contained in the data communicated to the honest-but-curious cloud during the collaborative learning process, Laplacian random noises are added to the communicated data. The proposed approach is applied to a case study of collaboratively learning a CNN for handwritten digit classification.

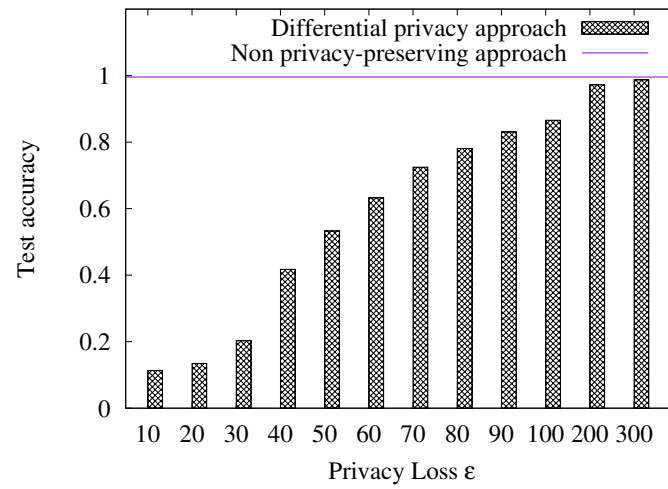


Fig. 5.5: Impact of privacy loss level ϵ on the test accuracy of the collaborative learning approach that perturbs the original data for DP.

Results show that collaboratively learned CNN with ϵ -DP has about 3% classification accuracy loss only, when the DP loss level ϵ is down to 2.

Chapter 6

Conclusion and Future Research

6.1 Conclusion

This thesis reviews the existing privacy-preserving ML approaches that were developed largely in the context of cloud computing and discusses their limitations in the context of IoT. Furthermore, this thesis proposes three different privacy preservation techniques for machine learning. First, in the proposed privacy-preserving inference approach, a small-scale non-linear transform in the form of a neural network is used to obfuscate data samples. Second, in the proposed lightweight privacy-preserving collaborative learning approach, the resource-constrained learning participants apply independent Gaussian projections on their training data samples and the coordinator applies deep learning to train a classifier based on the projected data. The above two approaches protect the confidentiality of the raw forms of the inference or training data transmitted to the honest-but-curious coordinator. In addition, the first approach can also protect specified private attributes contained in the inference data. Third, in the proposed differentially private collaborative learning approach, the different stages of a deep neural network are trained at the fog nodes and the cloud, respectively, using the training samples contributed by all the participating fog nodes. To protect the differential privacy contained in the data communicated to the honest-but-curious cloud during the collaborative learning process, Laplacian random noises are added to the communicated data. Extensive evaluation and implementation show the practicality and efficiency of the three proposed approaches.

In summary, this thesis adopts three different lightweight operations (neural obfuscation, Gaussian projection, and noise addition) on the two phases (training and inference) of the ML to preserve the privacy contained in the data samples. From the system perspective, the lightweight operations are more suitable on the resource-constraint IoT objects compared with the encryption-based approaches, e.g., homomorphic encryption. With the advance of the microchip and battery technologies, the IoT end devices may have compute capability and per-watt compute efficiency. Such a trend may enrich the design spaces of the low-power privacy preservation techniques for ML in IoT, which is of interest for the extensions of the approaches proposed in this thesis.

6.2 Future Research

6.2.1 Privacy-Preserving Unsupervised Learning

This thesis mainly focuses on the privacy-preserving supervised learning in the context of IoT. The proposed approaches solely consider a *classification* system consisting of a resourceful coordinator and many resource-constraint participants. However, the coordinator has the capability to provide unsupervised applications to the participants, for example, an aggregated clustering application, which is not considered in the aforementioned sections. Indeed, the unsupervised learning in the context of IoT is important as well.

Unsupervised learning uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition. Unsupervised learning provides an exploratory path to view data, allowing businesses to identify patterns in large volumes of data more quickly when compared to manual observation. Especially in the context of IoT, the labeling of the large volumes of data cost too much since the data are always the complex signals sensed from the diverse sensor. Thus, the labeling of the dataset is always artificially generated by the crowd [138], which bring huge cost for a well-labeling dataset.

Privacy-preserving unsupervised learning in the context of IoT is lacking. Autoencoder is one of the tools to achieve unsupervised learning. Although two recent studies [35,36] apply autoencoder to publish time series data (i.e., accelerator data) for inference, the training of their autoencoder still needs labeled data. An autoencoder leverages neural networks to compress data and then recreates a new representation of the original data's input. Therefore, applying the autoencoder on the distributed participants may bring the new representation for better analysis and novel application is feasible. Since the unlabeled data sensed from the IoT end device may contain privacy-sensitive information, designing privacy-preserving autoencoder is an open research problem.

6.2.2 Adversarial Protection Schemes

Chapter 3 involves adversarial learning framework into the proposed PriMask to enhance the privacy protection level. Within the adversarial learning framework, the PSP trains Attacknets as the adversary to make PriMask more robust against the potential privacy attacks. Intuitively, the information dispensable to the recognition task is likely to be removed along the adversary training direction [65]. The experiment results in this thesis show that it achieves satisfactory effectiveness in defending against the privacy attacks. Besides, the studies [65] and [66] apply adversarial learning to train the neural obfuscation, aiming at negating the adversary's capability of reconstructing the original data or extracting private attributes. Their results also show the robust defense against the malicious adversary. Therefore, adversarial learning can be a promising tool for developing various privacy protection schemes in the context of IoT.

However, utilizing the strength of adversarial learning framework in the context of IoT still faces some challenges. First, the adversarial learning process usually costs huge computation overhead due to the complex learning procedure. Thus, a proposed mechanism should consider the adversarial learning process on the resourceful backend for solving the related issues. Besides, in the adversarial learning framework, the resourceful backend plays the role of simulated malicious attacker. There, the defense against the corresponding attack in the adversarial learning performs well. However, the effect against other similar attack is not investigated. What's more, if the malicious attacker adopts the advanced attacknet which is more complex than the simulated attacknet, the

performance in the adversarial learning framework is not evaluated. The robustness of the adversarial learning in the context of IoT is an open research challenge.

References

- [1] J. Hamm, A. C. Champion, G. Chen, M. Belkin, and D. Xuan, “Crowd-ml: A privacy-preserving learning framework for a crowd of smart devices,” in *ICDCS*. IEEE, 2015, pp. 11–20.
- [2] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *CCS*. ACM, 2015, pp. 1310–1321.
- [3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” *arXiv preprint arXiv:1602.05629*, 2016.
- [4] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, and M. Ranzato, “Large scale distributed deep networks,” in *NIPS*, 2012, pp. 1223–1231.
- [5] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, “Parallelized stochastic gradient descent,” in *NIPS*, 2010, pp. 2595–2603.
- [6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.
- [7] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

REFERENCES

- [8] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, “Federated learning for mobile keyboard prediction,” *arXiv preprint arXiv:1811.03604*, 2018.
- [9] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *TCC*. Springer, 2006.
- [10] F. McSherry and K. Talwar, “Mechanism design via differential privacy,” in *FOCS*, vol. 7, 2007, pp. 94–103.
- [11] A. Roth and T. Roughgarden, “Interactive privacy via the median mechanism,” in *STOC*. ACM, 2010, pp. 765–774.
- [12] C. Dwork, “Differential privacy,” *Encyclopedia of Cryptography and Security*, pp. 338–340, 2011.
- [13] B. Liu, Y. Jiang, F. Sha, and R. Govindan, “Cloud-enabled privacy-preserving collaborative learning for mobile sensing,” in *SenSys*, 2012.
- [14] Y. Shen, C. Luo, D. Yin, H. Wen, R. Daniela, and W. Hu, “Privacy-preserving sparse representation classification in cloud-enabled mobile applications,” *Computer Networks*, vol. 133, pp. 59–72, 2018.
- [15] R. A. DeMillo, “Foundations of secure computation,” Georgia Institute of Technology, Tech. Rep., 1978.
- [16] T. Graepel, K. Lauter, and M. Naehrig, “MI confidential: Machine learning on encrypted data,” in *ICISC*. Springer, 2012, pp. 1–21.
- [17] J. Z. Zhan, L. Chang, and S. Matwin, “Privacy preserving k-nearest neighbor classification,” *Intl. J. Netw. Security*, vol. 1, no. 1, 2005.
- [18] Y. Qi and M. J. Atallah, “Efficient privacy-preserving k-nearest neighbor search,” in *ICDCS*. IEEE, 2008, pp. 311–319.
- [19] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, “Improving the robustness of deep neural networks via stability training,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4480–4488.

REFERENCES

- [20] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in *ICML*, 2016, pp. 201–210.
- [21] M. Barni, C. Orlandi, and A. Piva, “A privacy-preserving protocol for neural-network-based computation,” in *Proceedings of the 8th workshop on Multimedia and security*. ACM, 2006, pp. 146–151.
- [22] O. Goldreich, “Secure multi-party computation,” *Manuscript. Preliminary version*, vol. 78, 1998.
- [23] S. Servia-Rodríguez, L. Wang, J. R. Zhao, R. Mortier, and H. Haddadi, “Privacy-preserving personal model training,” in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018, pp. 153–164.
- [24] H. Brendan McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning differentially private recurrent language models,” *arXiv preprint arXiv:1710.06963*, 2017.
- [25] L. Sweeney, “k-anonymity: A model for protecting privacy,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.
- [26] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkatasubramanian, “l-diversity: Privacy beyond k-anonymity,” in *22nd International Conference on Data Engineering (ICDE’06)*. IEEE, 2006, pp. 24–24.
- [27] N. Li, T. Li, and S. Venkatasubramanian, “t-closeness: Privacy beyond k-anonymity and l-diversity,” in *2007 IEEE 23rd International Conference on Data Engineering*. IEEE, 2007, pp. 106–115.
- [28] R. Gross, L. Sweeney, F. De la Torre, and S. Baker, “Model-based face de-identification,” in *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW’06)*. IEEE, 2006, pp. 161–161.

REFERENCES

- [29] E. M. Newton, L. Sweeney, and B. Malin, “Preserving privacy by de-identifying face images,” *IEEE transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 232–243, 2005.
- [30] R. Gross, E. Airolidi, B. Malin, and L. Sweeney, “Integrating utility into face de-identification,” in *International Workshop on Privacy Enhancing Technologies*. Springer, 2005, pp. 227–242.
- [31] K. Chen and L. Liu, “Privacy preserving data classification with rotation perturbation,” in *Fifth IEEE International Conference on Data Mining (ICDM’05)*. IEEE, 2005, pp. 4–pp.
- [32] C. Huang, P. Kairouz, X. Chen, L. Sankar, and R. Rajagopal, “Context-aware generative adversarial privacy,” *Entropy*, vol. 19, no. 12, p. 656, 2017.
- [33] A. Rezaei, C. Xiao, J. Gao, B. Li, and S. Munir, “Application-driven privacy-preserving data publishing with correlated attributes,” in *Proceedings of the 2021 International Conference on Embedded Wireless Systems and Networks*, 2021, pp. 91–102.
- [34] G. Acs, L. Melis, C. Castelluccia, and E. De Cristofaro, “Differentially private mixture of generative neural networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 6, pp. 1109–1121, 2018.
- [35] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, “Mobile sensor data anonymization,” in *Proceedings of the international conference on internet of things design and implementation*, 2019, pp. 49–58.
- [36] O. Hajihassnai, O. Ardakanian, and H. Khazaei, “Obscurenet: Learning attribute-invariant latent representation for anonymizing sensor data,” in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 40–52.
- [37] Google, “Google’s edge tpu,” <https://cloud.google.com/edge-tpu/>, accessed: 2019-12-10.

- [38] G. Song and W. Chai, “Collaborative learning for deep neural networks,” in *Advances in Neural Information Processing Systems*, 2018, pp. 1832–1841.
- [39] U. M. Aïvodji, S. Gambs, and A. Martin, “Totfla: A secured and privacy-preserving smart home architecture implementing federated learning,” in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 175–180.
- [40] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy-preserving machine learning,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1175–1191.
- [41] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, “Privacy-preserving deep learning via additively homomorphic encryption,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018.
- [42] D. Byrd and A. Polychroniadou, “Differentially private secure multi-party computation for federated learning in financial applications,” *arXiv preprint arXiv:2010.05867*, 2020.
- [43] Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, and X. Zheng, “Privacy-preserving federated learning framework based on chained secure multiparty computing,” *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6178–6186, 2020.
- [44] V. Mugunthan, A. Polychroniadou, D. Byrd, and T. H. Balch, “Smpai: Secure multi-party computation for federated learning,” 2019.
- [45] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How to backdoor federated learning,” *arXiv preprint arXiv:1807.00459*, 2018.
- [46] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, “Analyzing federated learning through an adversarial lens,” *arXiv preprint arXiv:1811.12470*, 2018.
- [47] B. Hitaj, G. Ateniese, and F. Perez-Cruz, “Deep models under the gan: information leakage from collaborative deep learning,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 603–618.

REFERENCES

- [48] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [49] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.
- [50] I. J. Vergara-Laurens, L. G. Jaimes, and M. A. Labrador, “Privacy-preserving mechanisms for crowdsensing: Survey and research challenges,” *IEEE Internet of Things Journal*, vol. 4, no. 4, pp. 855–869, 2016.
- [51] L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks,” in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 41–47.
- [52] T. Zhu, G. Li, W. Zhou, and S. Y. Philip, “Differentially private data publishing and analysis: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1619–1638, 2017.
- [53] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.
- [54] K. Chaudhuri and C. Monteleoni, “Privacy-preserving logistic regression,” in *Advances in neural information processing systems*, 2009, pp. 289–296.
- [55] S. Song, K. Chaudhuri, and A. D. Sarwate, “Stochastic gradient descent with differentially private updates,” in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 245–248.
- [56] B. Jayaraman and D. Evans, “Evaluating differentially private machine learning in practice,” in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1895–1912.

- [57] K. Liu, H. Kargupta, and J. Ryan, “Random projection-based multiplicative data perturbation for privacy preserving distributed data mining,” *IEEE Transactions on knowledge and Data Engineering*, vol. 18, no. 1, pp. 92–106, 2005.
- [58] N. D. Lane and P. Georgiev, “Can deep learning revolutionize mobile sensing?” in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, 2015, pp. 117–122.
- [59] S. Yao, Y. Zhao, H. Shao, S. Liu, D. Liu, L. Su, and T. Abdelzaher, “Fastdeepiot: Towards understanding and optimizing neural network execution time on mobile and embedded devices,” in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, 2018, pp. 278–291.
- [60] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. V. Gool, “Ai benchmark: Running deep neural networks on android smartphones,” 2018.
- [61] E. Strubell, A. Ganesh, and A. McCallum, “Energy and policy considerations for deep learning in nlp,” 2019.
- [62] L. Jiang, R. Tan, X. Lou, and G. Lin, “On lightweight privacy-preserving collaborative learning for internet-of-things objects,” in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019, pp. 70–81.
- [63] S. A. Osia, A. S. Shamsabadi, A. Taheri, K. Katevas, S. Sajadmanesh, H. R. Rabiee, N. D. Lane, and H. Haddadi, “A hybrid deep learning architecture for privacy-preserving mobile analytics,” *arXiv preprint arXiv:1703.02952*, 2017.
- [64] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, “Not just privacy: Improving performance of private deep learning in mobile cloud,” in *KDD*. ACM, 2018, pp. 2407–2416.
- [65] S. Liu, J. Du, A. Shrivastava, and L. Zhong, “Privacy adversarial network: representation learning for mobile data privacy,” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 4, pp. 1–18, 2019.

REFERENCES

- [66] A. Li, J. Guo, H. Yang, F. D. Salim, and Y. Chen, “Deepobfuscator: Obfuscating intermediate representations with privacy-preserving adversarial learning on smartphones,” in *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 2021, pp. 28–39.
- [67] J. Chi, E. Owusu, X. Yin, T. Yu, W. Chan, P. Tague, and Y. Tian, “Privacy partitioning: Protecting user data during the deep learning inference phase,” *arXiv preprint arXiv:1812.02863*, 2018.
- [68] Z. He, T. Zhang, and R. B. Lee, “Model inversion attacks against collaborative inference,” in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 148–162.
- [69] —, “Attacking and protecting data privacy in edge–cloud collaborative inference systems,” *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9706–9716, 2020.
- [70] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, “Split learning for health: Distributed deep learning without sharing raw patient data,” *arXiv preprint arXiv:1812.00564*, 2018.
- [71] D. Ha, A. Dai, and Q. V. Le, “Hypernetworks,” *arXiv preprint arXiv:1609.09106*, 2016.
- [72] S. Chen, R. Jia, and G.-J. Qi, “Improved techniques for model inversion attacks,” 2020.
- [73] N. Carlini, S. Deng, S. Garg, S. Jha, S. Mahloujifar, M. Mahmoody, A. Thakurta, and F. Tramèr, “Is private learning possible with instance encoding?” in *IEEE Symposium on Security and Privacy (Oakland)*. IEEE, 2021, pp. 410–427.
- [74] “Mnist,” <http://yann.lecun.com/exdb/mnist/>, accessed: 2021-12-10.
- [75] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, “Adversarial machine learning,” in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, 2011, pp. 43–58.

REFERENCES

- [76] “PyTorch,” 2018, <https://pytorch.org/>.
- [77] “Pytorch mobile,” <https://pytorch.org/mobile/home/>.
- [78] “Jetson nano developer kit,” <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>.
- [79] J. Lu, G. Wang, and P. Moulin, “Human identity and gender recognition from gait sequences with arbitrary walking directions,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 1, pp. 51–61, 2013.
- [80] G. Garofalo, E. Argones Rúa, D. Preuveneers, W. Joosen *et al.*, “A systematic comparison of age and gender prediction on imu sensor-based gait traces,” *Sensors*, vol. 19, no. 13, p. 2945, 2019.
- [81] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones.” in *Esann*, vol. 3, 2013, p. 3.
- [82] CDC, “Distracted driving,” <https://bit.ly/3vNaIO9>.
- [83] Kaggle, <https://www.kaggle.com/c/state-farm-distracted-driver-detection>.
- [84] M. Maheer, “Python notebook using data from state farm distracted driver detection,” <https://www.kaggle.com/maheer/driver-distraction/notebook>.
- [85] Reuters, “Facebook critics want regulation, investigation after data misuse,” 2018, <https://reut.rs/2GwKF8p>.
- [86] J. Berr, “Equifax breach exposed data for 143 million consumers,” 2018, <https://cbsn.ws/2Qc8VOg>.
- [87] L. O’Donnell, “Zero-day flash exploit targeting middle east,” 2018, <https://threatpost.com/zero-day-flash-exploit-targeting-middle-east/132659/>.
- [88] A. Narayanan and V. Shmatikov, “How to break anonymity of the netflix prize dataset,” *arXiv preprint cs/0610105*, 2006.

REFERENCES

- [89] J. Hamm, A. Champion, G. Chen, M. Belkin, and D. Xuan, “Crowd-ml: A privacy-preserving learning framework for a crowd of smart devices,” in *Proc. ICDCS*. IEEE, 2015, pp. 11–20.
- [90] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *Proc. CCS*. ACM, 2015, pp. 1310–1321.
- [91] L. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, “Privacy-preserving deep learning via additively homomorphic encryption,” *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, 2018.
- [92] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy preserving machine learning,” in *Proc. CCS*. ACM, 2017, pp. 1175–1191.
- [93] H. Chabanne, A. de Wargny, J. Milgram, C. Morel, and E. Prouff, “Privacy-preserving classification on deep neural network,” *IACR Cryptology ePrint Archive*, vol. 2017, p. 35, 2017.
- [94] K. Liu, H. Kargupta, and J. Ryan, “Random projection-based multiplicative data perturbation for privacy preserving distributed data mining,” *IEEE Trans. knowl. Data Eng.*, vol. 18, no. 1, pp. 92–106, 2006.
- [95] Y. Rachlin and D. Baron, “The secrecy of compressed sensing measurements,” in *Proc. Allerton*. IEEE, 2008, pp. 813–817.
- [96] P. I. Wójcik and M. Kurdziel, “Training neural networks on high-dimensional data using random projection,” *Pattern Anal. Appl.*, pp. 1–11, 2018.
- [97] E. J. Candès *et al.*, “Compressive sampling,” in *Proceedings of the international congress of mathematicians*, vol. 3. Madrid, Spain, 2006, pp. 1433–1452.
- [98] R. Berinde, A. C. Gilbert, P. Indyk, H. Karloff, and M. J. Strauss, “Combining geometry and combinatorics: A unified approach to sparse signal recovery,” in *2008 46th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2008, pp. 798–805.

REFERENCES

- [99] G. Danezis and C. Diaz, “A survey of anonymous communication channels,” Microsoft Research, Tech. Rep., 2008, mSR-TR-2008-35.
- [100] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher, “DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework,” in *Proc. SenSys*. ACM, 2017, pp. 4:1–4:14.
- [101] L. N. Huynh, Y. Lee, and R. K. Balan, “Deepmon: Mobile gpu-based deep learning framework for continuous vision applications,” in *Proc. MobiSys*. ACM, 2017, pp. 82–95.
- [102] Google Cloud, “Edge TPU,” 2018, <https://cloud.google.com/edge-tpu/>.
- [103] C. Dwork, “Differential privacy,” in *Proc. ICALP*, 2006.
- [104] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, “Stealing machine learning models via prediction apis,” in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.
- [105] N. Ailon and B. Chazelle, “The fast johnson–lindenstrauss transform and approximate nearest neighbors,” *SIAM Journal on computing*, vol. 39, no. 1, pp. 302–322, 2009.
- [106] E. J. Candès and M. B. Wakin, “An introduction to compressive sampling,” *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, 2008.
- [107] S. Li, L. Da Xu, and X. Wang, “Compressed sensing signal and data acquisition in wireless sensor networks and internet of things,” *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2177–2186, 2013.
- [108] R. Tan, S.-Y. Chiu, H. H. Nguyen, D. K. Yau, and D. Jung, “A joint data compression and encryption approach for wireless energy auditing networks,” *ACM Trans. Sensor Networks*, vol. 13, no. 2, p. 9, 2017.
- [109] C. Wang, B. Zhang, K. Ren, and J. M. Roveda, “Privacy-assured outsourcing of image reconstruction service in cloud,” *IEEE Trans. Emerg. Topics Comput.*, vol. 1, no. 1, pp. 166–177, 2013.

REFERENCES

- [110] W. Xue, C. Luo, G. Lan, R. Rana, W. Hu, and A. Seneviratne, “Kryptein: a compressive-sensing-based encryption scheme for the internet of things,” in *Proc. IPSN*. IEEE, 2017, pp. 169–180.
- [111] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [112] A. Ben-Israel and T. N. Greville, *Generalized inverses: theory and applications*. Springer Science & Business Media, 2003, vol. 15.
- [113] C. C. Paige and M. A. Saunders, “Lsqqr: An algorithm for sparse linear equations and sparse least squares,” *ACM Transactions on Mathematical Software*, vol. 8, no. 1, pp. 43–71, 1982.
- [114] C.-C. Chang and C.-J. Lin, “Libsvm – a library for support vector machines,” 2018, <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- [115] M. Bierlaire, P. L. Toint, and D. Tuytens, “On iterative algorithms for linear least squares problems with bound constraints,” *Linear Algebra and its Applications*, vol. 143, pp. 111–143, 1991.
- [116] Z. Chen and J. J. Dongarra, “Condition numbers of gaussian random matrices,” *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 3, pp. 603–620, 2005.
- [117] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” *Conf. Theory of Cryptography*, pp. 265–284, 2006.
- [118] B. Bebersee, “Local differential privacy: a tutorial,” *arXiv preprint arXiv:1907.11908*, 2019.
- [119] Ú. Erlingsson, V. Pihur, and A. Korolova, “Rappor: Randomized aggregatable privacy-preserving ordinal response,” in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014, pp. 1054–1067.

REFERENCES

- [120] S. L. Warner, “Randomized response: A survey technique for eliminating evasive answer bias,” *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.
- [121] “source codes of the evaluation,” 2020, https://github.com/jls2007/TIOT_code.
- [122] Y. LeCun, C. Cortes, and C. J. Burges, “The mnist database of handwritten digits,” 2018, <http://yann.lecun.com/exdb/mnist/>.
- [123] “Spambase data set,” 2018, <https://archive.ics.uci.edu/ml/datasets/spambase>.
- [124] “free-spoken-digit-dataset,” 2019, <https://github.com/Jakobovski/free-spoken-digit-dataset>.
- [125] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.
- [126] B. Logan *et al.*, “Mel frequency cepstral coefficients for music modeling.” in *Ismir*, vol. 270, 2000, pp. 1–11.
- [127] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *Proc. ICLR*, 2015.
- [128] A. J. Bose and P. Aarabi, “Adversarial attacks on face detectors using neural net based constrained optimization,” in *Proc. Intl. Workshop Multimedia Signal Process.*, 2018.
- [129] G. Cormode, S. Jha, T. Kulkarni, N. Li, D. Srivastava, and T. Wang, “Privacy at scale: Local differential privacy in practice,” in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1655–1658.
- [130] D. Xu, M. Zheng, L. Jiang, C. Gu, R. Tan, and P. Cheng, “Lightweight and unobtrusive privacy preservation for remote inference via edge data obfuscation,” *arXiv preprint arXiv:1912.09859*, 2019.
- [131] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.

REFERENCES

- [132] “Raspberry Pi 2 Model B,” 2018, <https://bit.ly/1b75SRj>.
- [133] “Crowd-ml,” 2018, <https://github.com/jihunhamm/Crowd-ML>.
- [134] “Microsoft seal,” 2020, <https://www.microsoft.com/en-us/research/project/microsoft-seal/>.
- [135] M. Abadi, A. Chu, I. Goodfellow, H. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proc. CCS*. ACM, 2016, pp. 308–318.
- [136] C. Dwork, A. Roth *et al.*, “The algorithmic foundations of differential privacy,” *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.
- [137] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015, pp. 448–456.
- [138] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.