Differentially Private Federated Learning for Bandwidth and Energy Constrained Environments

Raouf Kerkouche

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Raouf KERKOUCHE

Thèse dirigée par **Claude CASTELLUCCIA**
et codirigée par **Pierre GENEVES**, CNRS

préparée au sein du **Laboratoire Institut National de Recherche en Informatique et en Automatique**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Apprentissage fédéré avec confidentialité différentielle pour les environnements contraints en bande passante et énergie

Differentially Private Federated Learning for Bandwidth and Energy Constrained Environments

Thèse soutenue publiquement le **7 juillet 2021**,
devant le jury composé de :

Monsieur EMILIANO DE CRISTOFARO
PROFESSEUR, University College London, Rapporteur
Monsieur REZA SHOKRI
PROFESSEUR ASSISTANT, National university of singapore, Rapporteur
Monsieur MASSIH-REZA AMINI
PROFESSEUR DES UNIVERSITES, UNIVERSITE GRENOBLE ALPES, Président
Monsieur AURELIEN BELLET
CHARGE DE RECHERCHE, INRIA CENTRE DE LILLE-NORD EUROPE, Examinateur
Monsieur MARC TOMMASI
PROFESSEUR DES UNIVERSITES, UNIVERSITE DE LILLE, Examinateur
Monsieur CLAUDE CASTELLUCCIA
DIRECTEUR DE RECHERCHE, INRIA CENTRE GRENOBLE-RHONE-ALPES, Directeur de thèse
Monsieur PIERRE GENEVES
DIRECTEUR DE RECHERCHE, CNRS DELEGATION ALPES, Co-directeur de thèse

Abstract

Machine Learning models are increasingly used in our daily life. For instance, these models can be used for content recommendation during a purchase or to help doctors while making medical decisions,etc. However, to obtain accurate and useful models, we generally need to train the models with large amount of data. Therefore, several entities with limited datasets may want to collaborate in order to improve their local model accuracy. In traditional machine learning, such collaboration requires to first store all entities' data on a centralized server before training the model on it. Such data centralization might be problematic when the data are sensitive and data privacy is required. Instead of sharing the training data, Federated Learning shares the model parameters between a server, which plays the role of aggregator, and the participating entities. More specifically, the server sends at each round the global model to some participants (downstream). These participants then update the received model with their local data and sends back the updated gradients' vector to the server (upstream). The server then aggregates all the participants' updates to obtain the new global model. This operation is repeated until the global model converges. Although Federated Learning improves both the privacy and the accuracy, it is not perfect. In fact, sharing gradients computed by individual parties can leak information about their private training data. Several recent attacks have demonstrated that a sufficiently skilled adversary, who can capture the model updates (gradients) sent by individual parties, can infer whether a specific record or a group property is present in the dataset of a specific party. Moreover, complete training samples can also be reconstructed purely from the captured gradients. Furthermore, Federated Learning is not only vulnerable to privacy attacks, it is also vulnerable to poisoning attacks which can drastically decrease the model accuracy. Finally, Federated Learning incurs large communication costs during upstream/downstream exchanges between the server and the parties. This can be problematic for applications based on bandwidth and energy-constrained devices, as it is the case for mobile systems, for instance. In this thesis, we first propose three bandwidth efficient schemes to reduce the bandwidth costs up to 99.9%. We then propose differentially private extensions of these schemes which are robust against honest-but-curious adversaries (server or participants) and protect the complete dataset of each participant (participant-level privacy). Moreover, our private solutions outperform standard privacy-preserving Federated Learning schemes in terms of accuracy and/or bandwidth efficiency. Finally, we investigate the robustness of our schemes against security attacks performed by malicious participants and discuss a possible privacy-robustness tradeoff which may spur further research.

Abstract (French)

En apprentissage automatique, plusieurs entités peuvent vouloir collaborer afin d'améliorer la précision de leur modèle local. Dans l'apprentissage automatique traditionnel, une telle collaboration nécessite de stocker d'abord les données de toutes les entités sur un serveur centralisé avant d'entraîner le modèle sur ces données. Cette centralisation des données peut s'avérer problématique lorsque les données sont sensibles et que leur confidentialité est requise. Au lieu de partager les données d'entraînement, l'apprentissage fédéré partage les paramètres du modèle entre un serveur, qui joue le rôle d'agrégateur, et les entités participantes. Plus précisément, le serveur envoie à chaque tour le modèle global à certains participants (en aval). Ces participants mettent ensuite à jour le modèle reçu avec leurs données locales et renvoient le vecteur des gradients mis à jour au serveur (en amont). Le serveur agrège alors toutes les mises à jour des participants pour obtenir le nouveau modèle global. Cette opération est répétée jusqu'à ce que le modèle global converge. Bien que l'apprentissage fédéré améliore la confidentialité, il n'est pas parfait. En effet, le partage des gradients calculés par les parties individuelles peut entraîner une fuite d'informations sur leurs données d'entraînement privées. Plusieurs attaques récentes ont démontré qu'un adversaire suffisamment habile, qui peut capturer les mises à jour du modèle (gradients) envoyées par les parties individuelles, peut déduire si une donnée spécifique ou une propriété de groupe est présent dans l'ensemble de données d'un participant. De plus, des échantillons d'entraînement complets peuvent également être reconstruits uniquement à partir des gradients capturés. En outre, l'apprentissage fédéré n'est pas seulement vulnérable aux attaques contre la vie privée, il est également vulnérable aux attaques par empoisonnement qui peuvent réduire considérablement la précision du modèle. Enfin, l'apprentissage fédéré entraîne des coûts de communication importants lors des échanges amont/aval entre le serveur et les parties. Cela peut être problématique pour les applications basées sur des dispositifs à bande passante et à énergie limitée comme c'est le cas pour les systèmes mobiles, par exemple. Dans cette thèse, nous proposons d'abord trois schémas efficaces en termes d'optimisation de la bande passante pour réduire les coûts jusqu'à 99,9 %. Ensuite, nous proposons une extension basée sur la confidentialité différentielle de nos schémas optimisés avec des garanties théoriques et qui surpassent en termes de précision le schéma standard d'apprentissage fédéré protégé avec la confidentialité différentielle. Enfin, nous étudions la robustesse de nos schémas contre les attaques de sécurité et nous discutons d'un compromis possible entre la confidentialité et la robustesse, ce qui pourrait ouvrir de nouvelles perspectives de recherches futures.

List of Tables

3.1 Summary of the attacks which are considered in our Security analysis introduced in Chapter 6. The description of all the attacks can be found there. 19

4.1 Summary of results using the FL-SIGN scheme. 25
4.2 Summary of results on Fashion-MNIST dataset using the FL-CS scheme. 29
4.3 Summary of results on Medical dataset using the FL-CS scheme. 29
4.4 Summary of results on Fashion-MNIST dataset using the FL-TOP scheme. 32
4.5 Summary of results on Medical dataset using the FL-TOP scheme. 32

5.1 Model accuracy and communication cost on MNIST dataset using FL-SIGN-DP. We give the communication cost per parameter value (bits/parameter) for any value of ε. 40
5.2 Model accuracy and communication cost with Fashion-MNIST dataset using FL-SIGN-DP. We give the communication cost per parameter value (bits/parameter) for any value of ε. 40
5.3 Summary of results on Fashion-MNIST dataset using the FL-CS-DP scheme. 45
5.4 Summary of results on Medical dataset using the FL-CS-DP scheme. 45
5.5 Summary of results on Fashion-MNIST dataset using the FL-TOP-DP scheme. 48
5.6 Summary of results on Medical dataset using the FL-TOP-DP scheme. 49

6.1 Random update attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the MNIST dataset depending on the fraction of malicious nodes. $\sigma_{\mathsf{Adv}} = 200$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge. 53
6.2 Random update attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the IMDB dataset depending on the fraction of malicious nodes. $\sigma_{\mathsf{Adv}} = 200$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge. 54
6.3 Gradient ascent attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the MNIST dataset. $\eta_{\mathsf{Adv}} = 10$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge. 54
6.4 Gradient ascent attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the IMDB dataset. $\eta_{\mathsf{Adv}} = 20$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge. 54
6.5 In-backdoor attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the MNIST dataset, η_{Adv} is set to 7, 3, 1 for 10%, 20%, 40% respectively. The table depicts the global accuracy convergence and the accuracy of the label "5" which is under attack. We choose the compression ratio r for FL-CS and FL-TOP such that each scheme reaches the same accuracy as FL-STD on the same settings but without the attack. Therefore, r is set to 5% and 0.5%, respectively. Note also, that we do not use boosting with FL-CS to not impact the sparsity of the model's update. 56

6.6 In-backdoor attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the CIFAR dataset, η_{Adv} is set to 7, 4, 2 for 10%, 20%, 40% respectively. The table depicts the global accuracy convergence and the accuracy of the label "airplane" which is under attack. We choose the compression ratio r for FL-CS and FL-TOP such that each scheme reaches the same accuracy as FL-STD on the same settings but without the attack. Therefore, r is set to 60% and 20%, respectively. Note also, that we do not use boosting with FL-CS to not impact the sparsity of the model's update. 56
6.7 Out-backdoors attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the MNIST dataset. η_{Adv} is set to 1 (no boosting). The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "0" to the targeted class "1" (attack accuracy). We choose the compression ratio r for FL-CS and FL-TOP such that each scheme reaches the same accuracy as FL-STD on the same settings but without the attack. Therefore, r is set to 5% and 0.5%, respectively. 57
6.8 Out-backdoors attack on FL-SIGN and FL-STD with the CIFAR dataset. η_{Adv} is set to 1 (no boosting). The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "airplane" to the targeted class "ship" (attack accuracy). We choose the compression ratio r for FL-CS and FL-TOP such that each scheme reaches the same accuracy as FL-STD on the same settings but without the attack. Therefore, r is set to 60% and 20%, respectively. 57
6.9 In-backdoor attack on FL-SIGN-DP and FL-SIGN with MNIST dataset. The table depicts the global accuracy convergence and the accuracy of the label "5" which is under attack. 58
6.10 In-backdoor attack on FL-SIGN and FL-SIGN-DP with Fashion-MNIST dataset. The table depicts the global accuracy convergence and the accuracy of the label "Sandal" which is under attack. 58
6.11 Out-backdoors attack on FL-SIGN-DP and FL-SIGN with MNIST dataset. The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "0" to the targeted class "1" (attack accuracy). 59
6.12 Out-backdoors attack on FL-SIGN-DP and FL-SIGN with Fashion-MNIST dataset. The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "T-shirt/Top" to the targeted class "Trouser" (attack accuracy). . . 59

8.1 Common environment and configuration of FL-SIGN, FL-CS, FL-TOP and FL-STD. $\gamma = 0.001$. 81
8.2 Common environment of the privacy part. $\gamma = 0.005$ and $T_{\text{cl}} = 200$. For FL-STD-DP, S is set to 1.73 and 2.15 for MNIST and Fashion-MNIST, respectively. For FL-SIGN-DP, S is fixed to $\sqrt{n}$. 82
8.3 Parameter Configuration for the Backdoor Attacks. FL-SIGN is used with the vote aggregation $\gamma = 0.001$. 82
8.4 Common environment between the schemes on Fashion-MNIST. ρ, η_G, λ and P are only used with FL-CS and FL-CS-DP. 82
8.5 Common environment between the schemes on the Medical dataset. ρ, η_G, λ and P are only used with FL-CS and FL-CS-DP. 83
8.6 Descriptions of features 83
8.7 Number of instances for our case study. The Medical dataset contains in total 1,271,733 records. 83
8.8 Common environment between the schemes. ρ, η_G and P are only used with FL-CS and FL-CS-DP. 83
8.9 Sensitivity S used for each scheme and for different compression ratio r on Fashion-MNIST. For FL-STD-DP, S is set to 2.40. 83
8.10 Sensitivity S used for each scheme and for different compression ratio r on the medical dataset. For FL-STD-DP, S is set to 1.40. 83
8.11 Summary of results on Fashion-MNIST dataset. 84
8.12 Summary of results on Medical dataset (Part 1). 85

8.13 Summary of results on Medical dataset (Part 2). 86

List of Figures

5.1 Distributions of the Top-K weight values (after convergence) for both FL-TOP and FL-STD schemes with the Fashion-MNIST dataset (left) and the medical dataset (right). 48

6.1 In-backdoor attack on FL-SIGN and FL-STD with the MNIST dataset, $\eta_{\text{Adv}} = 7$. The figure displays the global accuracy convergence and the accuracy of the label "5" which is under attack. 10% of the nodes are malicious. 55

6.2 In-backdoor attack on FL-SIGN and FL-STD with CIFAR dataset, $\eta_{\text{Adv}} = 7$. The figure displays the global accuracy convergence and the accuracy of the label "airplane" which is under attack. 10% of the nodes are malicious. 56

Acknowledgement

I take this opportunity to thank people and organizations without whom this dissertation would not be possible.

First, I would like to express my profound gratitude to my supervisors Claude Castelluccia and Pierre Genevès for their unwavering support, valuable advice and for offering me the flexibility to pursue my own research interests. I was very lucky to have them as supervisors. I would also like to thank Massih-Reza Amini, Emiliano De Cristofaro, Reza Shokri, Marc Tommasi and Aurélien Bellet for taking time to be part of my thesis committee.

I also want to thank the Grenoble Alpes Data Institute in the framework of the WP5 “Data Governance, Data Protection and Privacy” for awarding me fellowships to support my PhD.

I am very fortunate to have had the opportunity to collaborate with Gergely Ács with whom I learned and progressed a lot. Thank you very much for your availability, help and support.

I would also like to thank Mérouane Debbah who made me love the research world and who remains a model for me to follow. Thank you for everything you have done for me.

I would also like to thank the members of the Magnet team at Inria Lille, who hosted me at the beginning of my PhD to work on their decentralized architecture.

I am thankful to several people I met in the course of my PhD. Many thanks to Nabil Layaïda, Vincent Roca, Théodore Christakis, Cédric Lauradoux, Thomas Calmant, Mathieu Thiery, Supriya Sreekant Adhatarao, Belkacem Teibi, Coline Boniface, Amela Fejza, Muideen Adekunle Lawal, Sarah Chlyah, Laurent Carcone, Cécile Roisin for their many productive discussions and advise. I also acknowledge each and every member of the Privatics team, Tyrex team and the Chair on the Legal and Regulatory Implications of Artificial Intelligence. It has been such a great honour working alongside every one of you!

I would like to thank Microsoft for inviting me to present my research at the - Data Science Law Forum 3.0: "Operationalizing Responsible AI".

I would also like to thank my family and especially my parents, my brother, my sister, my aunt Fazia for their unconditional love and support. I owe a great deal to my family for providing me with lots of love and moral support.

I would like to thank my dear and loving wife Amira, who has given me much love and support during this adventure. You have shown patience and understanding and I will be forever indebted to you. You also gave me the most beautiful gift by giving birth to our little princess Léa whom we love so much.

Finally, I would like to thank my friends and everyone who helped me in one way or the other during my study. This achievement would not have been viable without all of you. Thank you all!

Dedication

To my dear and lovely little daughter Léa and wife Amira
To my beloved parents, brother Lyes and sister Doria.
To my dearest aunt Fazia who is like my second mom.

Contents

List of Tables v

List of Figures ix

Acknowledgement xiii

Dedication xv

1 Introduction 1
- 1.1 Motivation and Problem Statement . . . 1
- 1.2 Contributions . . . 2
  - 1.2.1 Bandwidth Efficient Federated Learning . . . 2
  - 1.2.2 Differentially Private and Bandwidth Efficient Federated Learning . . . 3
  - 1.2.3 Secure and Bandwidth Efficient Federated Learning . . . 4
- 1.3 Thesis Structure . . . 4
- 1.4 Publications . . . 5

I State-of-the-Art 7

2 Background 9
- 2.1 Federated Learning (FL-STD) . . . 9
- 2.2 Differential Privacy . . . 10
- 2.3 Compressive Sensing . . . 11
  - 2.3.1 Error Propagation . . . 12

3 Related Work 15
- 3.1 Machine Learning and Privacy . . . 15
  - 3.1.1 Inference Attacks . . . 15
  - 3.1.2 Privacy-Preserving Machine Learning . . . 16
- 3.2 Bandwidth Efficient Machine Learning . . . 17
- 3.3 Machine Learning and Security . . . 18
  - 3.3.1 Poisoning Attacks . . . 18
  - 3.3.2 Defenses . . . 19

II Contributions 21

4 Toward Bandwidth Efficient Federated Learning Schemes 23
- 4.1 Reducing Bandwidth by Compressing the updates . . . 24
  - 4.1.1 FL-SIGN: Bandwidth-Efficient Federated Learning via Quantization . . . 24
  - 4.1.2 Experimental Set-up . . . 25
  - 4.1.3 Performance Evaluation . . . 25
  - 4.1.4 FL-CS: Bandwidth-Efficient Federated Learning via Compressive Sensing . . . 25
  - 4.1.5 Experimental Set-up . . . 27

4.1.6 Performance Evaluation . . . 27
4.2 Reducing Bandwidth by Compressing the Model . . . 28
4.2.1 FL-TOP: Bandwidth-Efficient Federated Learning via Constraint . . . 28
4.2.2 Experimental Set-up . . . 31
4.2.3 Performance Evaluation . . . 31
4.3 Conclusion . . . 33

5 Designing Private and Bandwidth Efficient Federated Learning Schemes 35
5.1 Privacy Model . . . 36
5.2 FL-SIGN-DP: Private and Bandwidth-Efficient Federated Learning via Quantization . . . 36
5.2.1 Operation . . . 36
5.2.2 Discrete Gaussian Mechanism (DGM) . . . 37
5.2.3 Privacy of FL-SIGN-DP . . . 38
5.2.4 Communication Overhead . . . 38
5.2.5 Robustness of FL-SIGN-DP Against non-adversarial Client Failures . . . 39
5.2.6 Experimental Set-up . . . 39
5.2.7 Performance Evaluation . . . 40
5.3 FL-CS-DP: Private and Bandwidth-Efficient Federated Learning via Compressive Sensing 40
5.3.1 Operation . . . 40
5.3.2 Privacy of FL-CS-DP . . . 41
5.3.3 Robustness of FL-CS-DP Against non-adversarial Client Failures . . . 42
5.3.4 Experimental Set-up . . . 42
5.3.5 Results . . . 42
5.4 FL-TOP-DP: Private and Bandwidth-Efficient Federated Learning via Constraint . . . 44
5.4.1 Operation . . . 45
5.4.2 Privacy of FL-TOP-DP . . . 46
5.4.3 Remarks . . . 47
5.4.4 Robustness of FL-TOP-DP Against non-adversarial Client Failures . . . 47
5.4.5 Experimental Set-up . . . 47
5.4.6 Results . . . 48
5.5 Conclusion . . . 49

6 Security Analysis 51
6.1 Security Model . . . 52
6.1.1 Overall Model Degradation Attacks (Untargeted Attacks) . . . 52
6.1.1.1 Random Update Attack . . . 52
6.1.1.2 Gradient Ascent Attack . . . 52
6.1.2 Backdoor Attacks (Targeted Attacks) . . . 52
6.2 Security Analysis of FL-SIGN, FL-CS and FL-TOP . . . 53
6.2.1 Overall Model Degradation Attacks . . . 53
6.2.1.1 Random Update Attack . . . 53
6.2.1.2 Gradient Ascent Attack . . . 54
6.2.2 Backdoor Attacks . . . 54
6.2.2.1 In-backdoor Attack . . . 54
6.2.2.2 Out-backdoor Attack . . . 55
6.3 FL-SIGN-DP Security Analysis . . . 57
6.4 Conclusion . . . 58

7 Conclusion and perspectives 61
7.1 Concluding Remarks . . . 61
7.2 Future Research Directions . . . 62

Bibliography 65

8 Appendix **75**
8.1 Proofs for FL-SIGN and FL-SIGN-DP 75
8.1.1 Proof of Lemma 1 75
8.1.2 Proof of Theorem 2 75
8.1.3 Proof of Theorem 3 75
8.1.4 Convergence Proofs 76
8.2 Selection of Hyperparameters 78
8.3 Experimental Set-up 78
8.3.1 Model Architectures: 78
8.3.2 Datasets 79
8.4 Medical Data: Data Pre-Processing & Experimental Set-up Details 80
8.4.1 Medical Data: Data Pre-Processing 80
8.4.2 Medical Data: Performance Metrics 81
8.4.3 Evaluation Method. 81
8.5 Computational Environment 82

1 Introduction

> *AI is a tool. The choice about how it gets deployed is ours.*
>
> — **Oren Etzioni**
> (Professor of Computer Science, and CEO of the Allen Institute for Artificial Intelligence.)

1.1 Motivation and Problem Statement

During the last decade, Artificial Intelligence (AI) has gained in popularity and was used in various applications (e.g. medicine/HealthCare [APR19], ecology [For18], agriculture [Liu20], finance [Cao20]). In HealthCare, AI can be used, for example, to predict if a patient is more likely to be transferred to Intensive Care Unit or to die during his hospital stay. Such models are used by doctors to take better care of certain patients or to plan for enough patient beds per unit. Indeed, a recent study has shown a significant correlation between the availability of hospital resources (particularly ICU beds) and patient mortality during the early weeks of the COVID-19 pandemic [Uni21]. Therefore, doctors can save more patients' lives if they anticipate their influx per unit via those models.

Model training requires generally the availability of large datasets. In fact, there is often a correlation between the accuracy of the model and the data size. Professor Andrew Ng declared that "It's not who has the best algorithm that wins. It's who has the most data.". However, some data holder may not have enough data. For instance, rural hospitals have much less data than non-rural ones because they treat less patients.

Collaboration between users might be a solution to address this problem. In Machine Learning, different entities may want to collaborate in order to improve their local model accuracy. In traditional machine learning, such collaboration requires to first store all entities' data on a centralized server and then to train a model on it. However, data centralization is a problem when data are sensitive and privacy is required.

After the Facebook–Cambridge Analytica data scandal [Wik] where the British consulting firm Cambridge Analytica obtained and used the personal data of millions of Facebook users without their consent for political advertising, people have become more aware about the risk behind the use of their personal data. Fortunately, the European Union took the problem seriously long before this scandal. Indeed, it promulgated the General Data Protection Regulation (aka. GDPR [PC16]) a regulation in EU law on data protection and privacy in the European Union (EU) and the European Economic Area (EEA) to set the general rules for the use of sensitive and private data. Moreover, the specific characteristics of AI (e.g. opacity, complexity, dependency on data) which can negatively affect the fundamental rights defined in the EU Charter of Fundamental Rights [Uni12], has pushed the European Union to come up with a new proposal [PC21]. This proposal aims at enhancing and promoting the protection of the rights protected by the Charter, including respect for private life and protection of personal data (Articles 7 and 8 in [Uni12]).

In order to mitigate the privacy risk of data centralization, Federated Learning, which allows different entities to learn collaboratively a common model without sharing their data, was introduced [SS15; MMRHA16]. Instead of sharing the training data, Federated Learning shares the model parameters between a server, which plays the role of aggregator, and the participating entities. Therefore, it delegates the learning to the users from which data originate. Federated Learning has been gaining popularity and considered to train shared models for many applications such as input

text prediction, ad selection[1], drug discovery[2], or various medical applications [Cho+19] over the confidential data of many entities.

Although Federated Learning improves privacy, model parameters can still leak information about the training data. Indeed, [ZLH19; ZMB20; GBDM20] presented some attacks that allow an adversary to reconstruct parts of the training data of some participating entities. [NSH19; MSDS19] define membership attacks that allow one to infer if a particular record is included in the data of a specific entity. Similarly, [MSDS19] describes an attack which aims at inferring if a subgroup of people with a specific property, like for example skin color or ethnicity, is included in the dataset of a particular participating entity. Also, the property inference attack relates to properties that are uncorrelated to the main task.

A solution to prevent these attacks and provide theoretical guarantees is to use a privacy model called Differential Privacy [DR14]. Differential Privacy has been applied to federated learning in order to protect either each record included in the dataset of any entity (record-level guarantee), or the whole dataset of any entity (client-level guarantee). Unfortunately, it is well-known that Differential Privacy drastically degrades model accuracy as it requires adding random noise to the gradients (record-level) or to the updates (client-level) of each client.

Besides that, Federated Learning may be vulnerable against Poisoning attacks which includes Data Poisoning [BNL12; Rub+09; MZ15] or Model poisoning [BEGS17; BZAA18]. Data and Model poisoning attacks aim at reducing the model accuracy by altering respectively the training data or the learning process. Also, Poisoning attacks may either impact the global accuracy of the model (untargeted) or the accuracy of a specific class (targeted). Different solutions were proposed to be more robust against such attacks but they generally implies the access to each individual update. However, access to each individual update means that we can not use the secure aggregation protocol which allows to access only the aggregate update.

Finally, the communication cost between the server and the clients is another drawback of Federated Learning. Indeed, knowing that a model has on average millions of parameters, each of them encoded with 32 bits and each round requires an upstream and downstream communication cost, the total cost may be problematic. Therefore, some energy-constrained devices may suffer a lot during the training.

Medical and health datasets are considered as sensitive data. Indeed, Electronic Health Records datasets, for example, contains private information about different patients. Such private information may be harmful to the patient. An employer can decide to dismiss his employee if he knows that he is seriously ill, or, an insurance company may decide to increase the costs of the sick patient's insurance. On the other hand, these datasets are important and their benefits indisputable since they can help us analyzing and understanding interesting patterns. Various AI models are used for proactive healthcare management, disease mapping, finding effective medicines for diseases that are still not cured [Dal21; OWK21]. The main objective of this thesis is to design, implement and evaluate privacy-preserving Federated-learning algorithms. Another objective is to design robust solutions against state-of-the-art security attacks. Finally, the solutions should be bandwidth efficient and accurate for better usability and usefulness.

1.2 Contributions

In this section, we present our solutions and contributions that address the drawbacks of federated learning presented above.

1.2.1 Bandwidth Efficient Federated Learning

The first drawback of the federated learning scheme is the large communication costs incurred by upstream/downstream exchanges. We explored two different approaches: update compression and model compression. Update compression consists in compressing the size of the updates that are sent by the participants to the server. We proposed two "update compression" solutions, namely FL-SIGN and FL-CS. FL-SIGN is a quantization-based solution. It sends only a single bit per model parameter

[1] https://blog.chromium.org/2019/08/potential-uses-for-privacy-sandbox.html
[2] https://www.melloddy.eu

for aggregation. This extreme quantization reduces the required updates' bandwidth by a factor of 32, while still providing similar performance to the standard federated learning approach.

FL-CS for its part leverages the sparsity of the model updates and relies on the compressive sensing theory [Don06; CT06; CRT06] which enables to reconstruct a complete sparse data (ie,. image or signal) from few sampled points in it. More specifically, it loosely compresses the sparse model updates in federated learning using a slightly modified version of compressive sensing. Indeed, our protocol allows to save bandwidth and reduce communication costs by transferring only the low frequency components of the gradient vector to the server (instead of some random frequency components like in traditional compressive sensing). The server can reconstruct the approximated sparse gradient vector by efficiently solving a convex quadratic optimization problem. This approach provides more accurate reconstruction than simply applying the inverse Fourier transform on the low frequency components. Our approach is also scalable to large gradient vectors and is almost as accurate as the vanilla federated learning protocol, referred to FL-STD, without any compression, still incurring much smaller communication cost (up to 95% less).

Model compression consists in compressing the size of the model in order to reduce the number of element in the update vectors. We developed a solution, called FL-TOP, that harnesses the ability of the model to converge and to reach good accuracy under constraint. Indeed, in FL-TOP only some weights are updated while keeping all the remaining ones constant (defined as constraint). As all participants always update the same set of weights and transfer them, communication costs in both upstream and downstream are reduced. Our results show it also reduces the upstream and downstream bandwidth by a factor of 1000 compared to standard federated learning, which makes it particularly adapted to applications based on energy-constrained devices as it is the case for mobile systems.

1.2.2 Differentially Private and Bandwidth Efficient Federated Learning

We extended the aforementioned bandwidth efficient schemes to provide theoretical privacy guarantees. We proposed FL-SIGN-DP a privacy-preserving extension which provides client-level Differential Privacy (DP). Specifically, it hides any information that is unique to a client's training data, regardless whether it is about a single or multiple records, but still allows learning about characteristics that are common among multiple clients' training data. Our DP learning protocol, whose convergence rate is also computed analytically, produces models with an accuracy comparable to its non-private counterpart, even with stringent privacy guarantees (e.g., $\varepsilon = 1$). In order to diminish the communication costs of our DP algorithm, we proposed a novel discretized and distributed version of the Gaussian Mechanism. In particular, as opposed to the standard Gaussian Mechanism [DR14], the noise values come from a discretized domain and are tightly concentrated around its mean depending on the desired privacy guarantee ε. As a result, these values can be encoded with fewer bits than if they came from a continuous Gaussian distribution. Also, participants inject Gaussian noise in a distributed manner so that the sum of the noisy compressed vectors is differentially private. In addition, secure aggregation guarantees that the server (or any other third party) can only learn the noisy compressed aggregate.

The private extension of the standard federated learning scheme (FL-STD-DP) uses the full model which often has a high sensitivity. Unfortunately, the noise required by DP is proportional to its sensitivity. Furthermore, the accuracy of the model is inversely proportional to the added noise. Therefore, reducing sensitivity via bandwidth-efficient schemes can greatly improve model performance. The FL-CS-DP and FL-TOP-DP schemes reduce the sensitivity via compression and enable to add less noise, which improves the accuracy of the model compared to FL-STD-DP scheme.

The differentially private extension of FL-CS called FL-CS-DP adds Gaussian noise to the compressed gradients. In FL-CS-DP and similarly to FL-SIGN-DP, participants inject Gaussian noise in a distributed manner owing to the linear compression scheme. Reconstructing the approximated gradients is an instance of Basis Pursuit Denoising (or LASSO), which can be solved with efficient solvers that provide large accuracy despite the added Gaussian noise. We show that FL-CS-DP produces more accurate models than FL-STD-DP, that is, the differentially private variant of the vanilla federated learning

protocol without any compression. Therefore, compression boosts the accuracy of differentially private federated learning and also reduces bandwidth cost by more than 60% with early stopping [Cho+15c].

Also, we proposed a novel differentially private federated learning solution as an extension of FL-TOP called FL-TOP-DP that improves the model accuracy. The proposed scheme provides theoretical privacy guarantees, as it is based on Differential Privacy. Furthermore, it optimizes the model accuracy by constraining the model learning phase on a few selected weights. As all participants always update the same set of weights and transfer them to the server for aggregation, the proposal can be easily integrated with secure aggregation [Bon+16], exactly as for FL-SIGN-DP and FL-CS-DP, which allows parties to add less noise than other decentralized perturbation approaches such as randomized response [EPK14] used in local differential privacy. Finally, we noticed that, compression (bandwidth efficiency) which reduces the sensitivity of the models in FL-CS-DP and FL-TOP-DP seems to be compatible with differential privacy. Indeed, the two combined improve the utility of the private model.

Finally, in FL-CS-DP and FL-TOP-DP, we add the noise to specific coordinates that often have much larger update values than others. The objective is to increase the value-to-noise ratio to achieve better performance. In FL-CS-DP, the noise is added to the first coefficients. Indeed, due to the large energy compaction property of DCT, the first coefficients, which correspond to the low frequency components of the update vector, tend to have the largest magnitude and hence convey the most information about the model update. Similarly, the constraint defined in FL-TOP-DP by updating only the Top-K weights, leads to the increase of update values of those weights. Therefore, adding the noise to the updates of the Top-K weights leads to increase the value-to-noise level.

1.2.3 Secure and Bandwidth Efficient Federated Learning

The concept of "curse of dimensionality" was introduced in [CSSH19]. The authors claim that large models are more vulnerable to privacy and security attacks. Therefore, reducing the model size (or its update) should make the model more robust against such attacks.

To validate this assumption, we experimentally evaluated the robustness of our bandwidth efficient schemes by implementing and testing several State-of-the-Art security attacks, such as model degradation (where the adversary aims at reducing the global model accuracy) or backdoor inclusion attacks (where the adversary aims at inserting hidden backdoors). We showed that, FL-SIGN, is more resilient to these attacks than the other scheme thanks to: (1) the quantization which limits the adversarial impact on the aggregate as it requires to bound each parameter's update with +1/-1. Therefore, the boosting which is commonly used by an adversary to surpass the updates of honest participants can be easily detected. (2) the majority vote (by taking the median) makes also our approach more robust against poisoning attacks. Indeed, if we assume that we have more honest participants than malicious ones, we are more likely to update the weights correctly.

Although, FL-SIGN is robust against security attacks and may suggest a possible compatibility between compression and robustness to security attacks, both FL-CS and FL-TOP are vulnerable against poisoning attacks. Therefore, compression does not necessarily imply robustness against security attacks.

Finally, we decided to evaluate the robustness of FL-SIGN-DP against poisoning attacks. Although FL-SIGN is robust against various attacks, its differentially private variant FL-SIGN-DP however turns out to be more vulnerable to the security attacks. Indeed, the attacks are inherently concealed by the noise which is introduced to guarantee Differential Privacy. Our result suggests a possible privacy-security tradeoff.

1.3 Thesis Structure

The thesis is structured as follows. Section 2 details the preliminaries including the basic Federated Learning algorithm (FL-STD) and Differential Privacy (DP). Section 3 introduces related work. To address the large bandwidth costs drawback of Federated Learning, we define in Chapter 4 three bandwidth efficient schemes, namely FL-SIGN, FL-CS and FL-TOP which are as accurate FL-STD. We therefore propose in Chapter 5 to extend theses schemes in order to obtain differential private guarantees and defined as FL-SIGN-DP, FL-CS-DP and FL-TOP-DP, respectively, as a solution to privacy

issues in FL-STD. In Chapter 6 we investigate the robustness of the proposed schemes on state-of-the-art security attacks, and finally Section 7 provides a summary and additional discussions about the proposed algorithms and future works.

1.4 Publications

My publications written during my thesis are listed below:

– "*Constrained Differentially Private Federated Learning for Low-bandwidth Devices*",
Raouf Kerkouche, Gergely Ács, Claude Castelluccia and Pierre Genevès.
To appear in the Conference on Uncertainty in Artificial Intelligence (UAI'21), [PDF].
Abstract. Federated learning becomes a prominent approach when different entities want to learn collaboratively a common model without sharing their training data. However, Federated learning has two main drawbacks. First, it is quite bandwidth inefficient as it involves a lot of message exchanges between the aggregating server and the participating entities. This bandwidth and corresponding processing costs could be prohibitive if the participating entities are, for example, mobile devices. Furthermore, although federated learning improves privacy by not sharing data, recent attacks have shown that it still leaks information about the training data. This paper presents a novel privacy-preserving federated learning scheme. The proposed scheme provides theoretical privacy guarantees, as it is based on Differential Privacy. Furthermore, it optimizes the model accuracy by constraining the model learning phase on few selected weights. Finally, as shown experimentally, it reduces the upstream and downstream bandwidth by up to 99.9% compared to standard federated learning, making it practical for mobile systems.

– "*Compression Boosts Differentially Private Federated Learning*",
Raouf Kerkouche, Gergely Ács, Claude Castelluccia and Pierre Genevès.
To appear in the 6th IEEE European Symposium on Security and Privacy (Euro S&P'21), [PDF].
Abstract. Federated Learning allows distributed entities to train a common model collaboratively without sharing their own data. Although it prevents data collection and aggregation by exchanging only parameter updates, it remains vulnerable to various inference and reconstruction attacks where a malicious entity can learn private information about the participants' training data from the captured gradients. Differential Privacy is used to obtain theoretically sound privacy guarantees against such inference attacks by noising the exchanged update vectors. However, the added noise is proportional to the model size which can be very large with modern neural networks. This can result in poor model quality. In this paper, compressive sensing is used to reduce the model size and hence increase model quality without sacrificing privacy. We show experimentally, using 2 datasets, that our privacy-preserving proposal can reduce the communication costs by up to 95% with only a negligible performance penalty compared to traditional non-private federated learning schemes.

– "*Privacy-Preserving and Bandwidth-Efficient Federated Learning: An Application to In-Hospital Mortality Prediction*",
Raouf Kerkouche, Gergely Ács, Claude Castelluccia and Pierre Genevès.
ACM Conference on Health, Inference and Learning (CHIL'21), [PDF].
Abstract. Machine Learning, and in particular Federated Machine Learning, opens new perspectives in terms of medical research and patient care. Although Federated Machine Learning improves over centralized Machine Learning in terms of privacy, it does not provide provable privacy guarantees. Furthermore, Federated Machine Learning is quite expensive in terms of bandwidth consumption as it requires participant nodes to regularly exchange large updates. This paper proposes a bandwidth-efficient privacy-preserving Federated Learning that provides theoretical privacy guarantees based on Differential Privacy. We experimentally evaluate our proposal for in-hospital mortality prediction using a real dataset, containing Electronic Health Records of about

one million patients. Our results suggest that strong and provable patient-level privacy can be enforced at the expense of only a moderate loss of prediction accuracy.

– "*Federated Learning in Adversarial Settings*",
Raouf Kerkouche, Gergely Ács, Claude Castelluccia.
Submitted for publication, [PDF].
Abstract. Federated Learning enables entities to collaboratively learn a shared prediction model while keeping their training data locally. It prevents data collection and aggregation and, therefore, mitigates the associated privacy risks. However, it still remains vulnerable to various security attacks where malicious participants aim at degrading the generated model, inserting backdoors, or inferring other participants' training data. This paper presents a new federated learning scheme that provides different trade-offs between robustness, privacy, bandwidth efficiency, and model accuracy. Our scheme uses biased quantization of model updates and hence is bandwidth efficient. It is also robust against state-of-the-art backdoor as well as model degradation attacks even when a large proportion of the participant nodes are malicious. We propose a practical differentially private extension of this scheme which protects the whole dataset of participating entities. We show that this extension performs as efficiently as the non-private but robust scheme, even with stringent privacy requirements but are less robust against model degradation and backdoor attacks. This suggests a possible fundamental trade-off between Differential Privacy and robustness.

Part I

State-of-the-Art

2 Background

2.1 Federated Learning (FL-STD)

In federated learning [SS15; MMRHA16], multiple parties (clients) build a common machine learning model from the union of their training data without sharing them. At each round of the training, a set of selected clients retrieve the global model from the parameter server, update the global model based on their own training data, and send back their updated model to the server. The server aggregates the received updated models to obtain a global model that is re-distributed to some other parties in the next round.

More specifically, a subset $\mathbb{K}$ of all N clients are randomly selected at each round to update the global model such that $|\mathbb{K}| = C \times N$, where C denotes the fraction of selected clients. At round t, a selected client $k \in \mathbb{K}$ executes T_{gd} local gradient descent iterations on the common model $\mathbf{w}_{t-1}$ using its own training data D_k ($D = \cup_{k \in \mathbb{K}} D_k$), and obtains the updated model $\mathbf{w}_t^k$, where the number of weights is denoted by n (i.e., $|\mathbf{w}_t^k| = |\Delta \mathbf{w}_t^k| = n$ for all k and t). Each client k submits the update $\Delta \mathbf{w}_t^k = \mathbf{w}_t^k - \mathbf{w}_{t-1}^k$ to the server, which then updates the common model as follows: $\mathbf{w}_t = \mathbf{w}_{t-1} + \sum_{k \in \mathbb{K}} \frac{|D_k|}{\sum_j |D_j|} \Delta \mathbf{w}_t^k$, where $|D_k|$ is known to the server for all k (a client's update is weighted with the size of its training data). The server stops training after a fixed number of rounds T_{cl}, or when the performance of the common model does not improve on a held-out data.

Note that each D_k may be generated from different distributions (i.e., Non-IID case), that is, any client's local dataset may not be representative of the population distribution [MMRHA16]. This can happen, for example, when not all output classes are represented in every client's training data. The federated learning of neural networks is summarized in Alg. 1. In the sequel, each client is assumed to use the same model architecture.

Algorithm 1: FL-STD: Federated Learning

Server:
  Initialize common model $\mathbf{w}_0$
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\Delta \mathbf{w}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \sum_k \frac{|D_k|}{\sum_j |D_j|} \Delta \mathbf{w}_t^k$
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\mathbf{w}_{t-1}^k)$**:**
  $\mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
  Output: Model update $(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k)$

The motivation of federated learning is three-fold: first, it aims to provide confidentiality of each participant's training data by sharing only model updates instead of potentially sensitive training data. Second, in order to decrease communication costs, clients can perform multiple local SGD iterations before sending their update back to the server. Third, at each round, only a few clients are required to perform local training of the common model, which further decreases communication costs and makes the approach especially appealing with large number of clients.

Algorithm 2: Stochastic Gradient Descent

Input: D : training data, T_{gd} : local epochs, $\mathbf{w}$: weights
for $t = 1$ **to** T_{gd} **do**
    Select batch $\mathbb{B}$ from D randomly
    $\mathbf{w} = \mathbf{w} - \eta \nabla f(\mathbb{B}; \mathbf{w})$
end
Output: Model $\mathbf{w}$

There is different types of federated learning schemes which can be classified based on the distributions of data features and data samples among participants [YLCT19]. The most used is the horizontally federated learning (HFL) which assumes that participants share similar features but they differ in users' records [KC04; SS15; MMRHA16]. The second approach is called vertically federated learning (VFL) and it assumes that participants share the same users' records but with different features [VC02]. Federated transfer learning (FTL) is another approach where participants have little overlap in both users' records and features [YLCT19]. Similarly, federated learning can be classified based on the model architectures among participants. Generally, we assume that all participants train models with the same architecture and it is defined as Federated Learning with Homogeneous Architectures. However, in federated learning with Heterogeneous Architectures each participant can use a unique model architecture over the training process [LW19].

Standard Federated Learning requires a single server which aggregates the received models/updates from participants. However, in Decentralized Federated Learning, each participant may play the role of the server at a given round [YLCT19; Lyu+19; LYY20]. Therefore, the number of possible server during the training process can be equal to the number of participants.

2.2 Differential Privacy

Differential privacy allows a party to privately release information about a dataset: a function of an input dataset is perturbed, so that any information which can differentiate a record from the rest of the dataset is bounded [DR14].

Definition 1 (Privacy loss). *Let $\mathcal{A}$ be a privacy mechanism which assigns a value $Range(\mathcal{A})$ to a dataset D. The privacy loss of $\mathcal{A}$ with datasets D and D' at output $O \in Range(\mathcal{A})$ is a random variable $\mathcal{P}(\mathcal{A}, D, D', O) = \log \frac{\Pr[\mathcal{A}(D)=O]}{\Pr[\mathcal{A}(D')=O]}$ where the probability is taken on the randomness of $\mathcal{A}$.*

Definition 2 ((ϵ, δ)-Differential Privacy [DR14]). *A privacy mechanism $\mathcal{A}$ guarantees (ε, δ)-differential privacy if for any database D and D', differing on at most one record, $\Pr_{O \sim \mathcal{A}(D)}[\mathcal{P}(\mathcal{A}, D, D', O) > \varepsilon] \leq \delta$.*

Intuitively, this guarantees that an adversary, provided with the output of $\mathcal{A}$, can draw almost the same conclusions (up to ε with probability larger than $1 - \delta$) about any record no matter if it is included in the input of $\mathcal{A}$ or not [DR14]. That is, for any record owner, a privacy breach is unlikely to be due to its participation in the dataset.

Moments Accountant. Differential privacy maintains composition; the privacy guarantee of the k-fold adaptive composition of $\mathcal{A}_{1:k} = \mathcal{A}_1, \ldots, \mathcal{A}_k$ can be computed using the moments accountant method [Aba+16]. In particular, it follows from Markov's inequality that $\Pr[\mathcal{P}(\mathcal{A}, D, D', O) \geq \varepsilon] \leq \mathbb{E}[\exp(\lambda \mathcal{P}(\mathcal{A}, D, D', O))] / \exp(\lambda \varepsilon)$ for any output $O \in Range(\mathcal{A})$ and $\lambda > 0$. This implies that $\mathcal{A}$ is (ε, δ)-DP with $\delta = \min_\lambda \exp(\alpha_{\mathcal{A}}(\lambda) - \lambda \varepsilon)$, where $\alpha_{\mathcal{A}}(\lambda) = \max_{D,D'} \log \mathbb{E}_{O \sim \mathcal{A}(D)}[\exp(\lambda \mathcal{P}(\mathcal{A}, D, D', O))]$ is the log of the moment generating function of the privacy loss. The privacy guarantee of the composite mechanism $\mathcal{A}_{1:k}$ can be computed using that $\alpha_{\mathcal{A}_{1:k}}(\lambda) \leq \sum_{i=1}^{k} \alpha_{\mathcal{A}_i}(\lambda)$ [Aba+16].

Gaussian Mechanism. There are a few ways to achieve DP, including the Gaussian mechanism [DR14]. A fundamental concept of all of them is the *global sensitivity* of a function [DR14].

Definition 3 (Global L_p-sensitivity). *For any function $f : \mathcal{D} \to \mathbb{R}^n$, the L_p-sensitivity of f is $\Delta_p f = \max_{D,D'} ||f(D) - f(D')||_p$, for all D, D' differing in at most one record, where $||\cdot||_p$ denotes the L_p-norm.*

The Gaussian Mechanism [DR14] consists of adding Gaussian noise to the true output of a function. In particular, for any function $f : \mathcal{D} \to \mathbb{R}^n$, the Gaussian mechanism is defined as adding i.i.d Gaussian noise with variance $(\Delta_2 f \cdot \sigma)^2$ and zero mean to each coordinate value of $f(D)$. Recall that the pdf of the Gaussian distribution with mean μ and variance ξ^2 is

$$\mathsf{pdf}_{\mathcal{G}(\mu,\xi)}(x) = \frac{1}{\sqrt{2\pi}\xi} \exp\left(-\frac{(x-\mu)^2}{2\xi^2}\right) \tag{2.1}$$

In fact, the Gaussian mechanism draws vector values from a multivariate spherical (or isotropic) Gaussian distribution which is described by random variable $\mathcal{G}(f(D), \Delta_2 f \cdot \sigma \mathbf{I}_n)$, where n is omitted if it's unambiguous in the given context.

2.3 Compressive Sensing

Compressive Sensing (CS) [Don06; CT06; CRT06] aims to reconstruct the original signal from significantly fewer samples (or measurements) than other traditional sampling techniques, which are based on the Nyquist-Shannon theorem, by exploiting the sparsity of the signal.

Consider a signal $\mathbf{x} \in \mathbb{R}^n$ which admits a sparse representation $\mathbf{s} \in \mathbb{R}^n$, that is, there exists a *sparsity orthonormal basis* with matrix $\Psi \in \mathbb{R}^{n \times n}$ such that:

$$\mathbf{x} = \Psi \mathbf{s} \tag{2.2}$$

Here, $\mathbf{s}$ is U-sparse if $\|\mathbf{s}\|_0 = \mathbf{U}$. Ψ can denote any linear transformation, such as Discrete Fourier/Cosine or Wavelet Transform, which renders the original signal $\mathbf{x}$ sparse. If $\mathbf{x}$ is already sparse, then Ψ can be the identity matrix which corresponds to the canonical sparsity basis.

In CS, $\mathbf{x}$ is reconstructed from some of its *linear measurements*. For m measurements, the signal is "sampled" in m values $\mathbf{y}_j = \langle \phi_j, \mathbf{x} \rangle$ $(1 \leq j \leq m)$, where the vectors $\phi_j \in \mathbb{R}^n$ constitute the sensing basis matrix $\Phi = (\phi_1, \phi_2, \ldots, \phi_m)^\top \in \mathbb{R}^{m \times n}$. Here, $m = r \times n$, where r is the compression ratio. Therefore, the compression operator $\mathcal{C}$ is defined as:

$$\mathcal{C}(\mathbf{x}, m) = \mathbf{y} = \Phi \mathbf{x} = \Phi \Psi \mathbf{s} = \Theta \mathbf{s} \tag{2.3}$$

where Θ is the sparsity sensing matrix.

There are several options to select the sensing matrix Φ. When Φ is a random matrix (e.g., each element of Φ is an iid sample from $\mathcal{G}(0, 1/m)$), then Ψ works well with an arbitrary sparsity basis [JV10]. On the other hand, the numerical reconstruction of $\mathbf{x}$ in that case has a complexity of $O(mn)$ which can be very large (recall that n is the model size in the order of 10^6). Another (faster) option for the sensing matrix Φ is when it is composed of random m rows of the matrix of the (real) Discrete Fourier/Cosine Transform. Then, matrix multiplication can be executed with the Fast Fourier Transform (FFT) in $O(n \ln n)$, but such sensing matrix provides accurate reconstruction if Ψ is the identity matrix, i.e. $\mathbf{x}$ is already sparse [JV10]. Fortunately, this usually holds for gradient vectors (or can be made as such by sparsification without significantly affecting convergence) and hence we will use this option in this dissertation.

In order to recover $\mathbf{s}$ from $\mathbf{y}$, one has to solve a system of linear equations with m equations and n unknowns. Although this system seems underdetermined because $m < n$, CS exploits the U-sparsity of $\mathbf{s}$ for the reconstruction. It aims to reconstruct the sparse vector $\mathbf{s}$ from $\mathbf{y} = \Theta \mathbf{s}$ given the sparsity sensing basis Θ by solving the following optimization problem:

$$\arg\min_s \|\mathbf{s}\|_0 \quad \text{s.t.} \quad \mathbf{y} = \Theta \mathbf{s}$$

Since this optimization problem is NP-complete [Nat95; JV10], it is further relaxed into the following slightly different problem called Basis Pursuit (BP) [CDS01]:

$$\arg\min_s \|\mathbf{s}\|_1 \quad \text{s.t.} \quad \mathbf{y} = \Theta \mathbf{s}$$

Indeed, the convex L_1-norm usually approximates the non-convex L_0-norm well, and the relaxed optimization problem can be efficiently solved with any convex optimization technique [JV10] (e.g., with an LP solver).

When the measurements $\mathbf{y}$ are noisy (i.e., $\mathbf{y} = \Theta\mathbf{s} + \mathbf{z}$, where $\mathbf{z} \in \mathbb{R}^m$ is the additional bounded iid noise, i.e. $\|\mathbf{z}\|_2 \leq \kappa$), then the following convex quadratic variant of BP called Basis Pursuit Denoising (BPDN) is rather considered:

$$\mathcal{R}(\mathbf{y}, \kappa) = \arg\min_s \|\mathbf{s}\|_1 \quad \text{s.t.} \quad \|\mathbf{y} - \Theta\mathbf{s}\|_2 \leq \kappa$$

and therefore

$$\mathcal{D}(\mathbf{y}, n) = \Psi\left(\arg\min_s \frac{1}{2}\|\mathbf{y} - \Theta\mathbf{s}\|_2^2 + \lambda\|\mathbf{s}\|_1\right), \tag{2.4}$$

Eq. (2.4) defines our decompression operator and is an instance of convex quadratic programming. In this paper, we use the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) algorithm [AG07; Tay20], an extension of Limited-memory BFGS, which is a numerical scalable optimization procedure that can efficiently solve Eq. (2.4).

When $\lambda \to 0$, the problem in Eq. (2.4) becomes BP because $\lambda\|\mathbf{s}\|_1$ tends to 0. In the case of non-noisy sensing measurements, a BP decoder is more adapted to reconstruct the sparse signal s. Otherwise, BPDN is more suited. This has particular importance in our case when the compressed vector (measurements) are noised to guarantee Differential Privacy, i.e., $\mathbf{y} = \Theta\mathbf{s}+\mathbf{z}$ where $\mathbf{z} \sim \mathcal{N}(0, S\mathbf{I}\sigma)$ (see Section 5.3.1). Approximate signal reconstruction from noisy measurements have been theoretically justified in [Can08] from the Restricted Isometry Property of Θ.

Definition 4 (Restricted Isometry Property (RIP) [CT05])**.** *The U-restricted isometry constant $0 \leq \delta_U < 1$ of a matrix $\Theta \in \mathbf{R}^{m \times n}$ is defined as the smallest number such that:*

$$(1 - \delta_U)\|\mathbf{s}\|_2^2 \leq \|\Theta\mathbf{s}\|_2^2 \leq (1 + \delta_U)\|\mathbf{s}\|_2^2$$

for all U-sparse vector $\mathbf{s} \in \mathbb{R}^n$ and we say that the matrix Θ obeys the Restricted Isometry Property (or RIP(U,δ_U)) of order $U < m$.

Theorem 1 (Reconstruction error of BPDN [Can08])**.** *If Θ is RIP$(2U, \delta_U)$ and $\delta_U < \sqrt{2} - 1$, then $||\mathbf{s} - \mathcal{R}(\mathbf{y}, \kappa)||_2 \leq C\kappa + (D/\sqrt{K})||\mathbf{s} - \mathbf{s}_K||_1$, where C and D are constants and $\mathbf{s}_K$ is a vector with all but the K-largest entries of $\mathbf{s}$ set to zero*[1].

Finally, notice that the compression operator $\mathcal{C}$ in Eq. (2.3) is *linear*, which means that:

$$\sum_i \mathcal{C}(\mathbf{x}_i, m) = \mathcal{C}\left(\sum_i \mathbf{x}_i, m\right)$$

and therefore

$$\mathcal{D}\left(\sum_i \mathcal{C}(\mathbf{x}_i, m)\right) \approx \sum_i \mathbf{x}_i$$

This linearity allows to combine secure aggregation and compressive sensing described in Section 5.3.1.

2.3.1 Error Propagation

Biased estimation of the gradients may prevent model convergence unless the approximation error introduced by lossy compression techniques, such as compressive sensing, sketching, or quantization, is accumulated and re-injected in every optimization round [KRSJ19] as follows:

[1] For instance, for $\delta_U = 0.2$, $C < 4.2$ and $D < 8.5$

$$
\begin{aligned}
\mathbf{g}_t &= \nabla f(\mathbb{B}, \mathbf{w}_{t-1}) : \text{Computing gradients on batch } \mathbb{B} \\
\mathbf{p}_t &= \eta \mathbf{g}_t + \mathbf{e}_{t-1} : \text{Error feedback (correction)} \\
\Delta_t &= \mathcal{D}(\mathcal{C}(\mathbf{p}_t)) : \text{Reconstruction of } \mathbf{p}_t \\
\mathbf{w}_t &= \mathbf{w}_{t-1} - \Delta_t : \text{Updating model parameters (weights)} \\
\mathbf{e}_t &= \mathbf{p}_t - \Delta_t : \text{Error accumulation}
\end{aligned}
$$

The corrected direction $\mathbf{p}_t$ is obtained by adding the error $\mathbf{e}_{t-1}$ accumulated over all iterations to $\mathbf{g}_t$ (see Alg. 2 in [KRSJ19] for more details). Here, the error is calculated based on the biased estimation of the update given by $\mathcal{D}(\mathcal{C}(\mathbf{p}_t))$.

3 Related Work

3.1 Machine Learning and Privacy

3.1.1 Inference Attacks

There exists a few inference attacks specifically designed against federated learning schemes. In property inference attack [MSDS19], the adversary's goal is to infer whether records with a specific property are included in the training dataset of the other participants (called batch property inference). The authors demonstrate the attack by inferring whether black people are included in any of the training datasets, where the common model is trained for gender classification (i.e., the inferred property is independent of the learning objective). The attack assumes the participation of all the participants at each round including the adversary. Therefore, the adversary can retrieve the aggregated update of honest participants by first computing the difference between the received model at round t and $t-1$ and then by subtracting his update sent to the server at round $t-1$. It can also be passive or active. In passive mode, the adversary locally "emulate" the collaborative training process by sampling data with and without the property, calculate the aggregate of honest updates and train a binary classifier to distinguish between the aggregates which includes an update based on the data with and without the property. However, the attack is more powerful when the adversary is active and use multi-task learning to simultaneously optimize the main task and recognize batch properties. As a result, the malicious update will lead the global model to learn a separable representation of the gradients generated by using data with and without the property and therefore enabling the adversary to decide whether the training data has the property.

An adversary who has a black-box access to a classifier model (has a view only on the outputs of the last layer), can use model inversion attacks [FJR15] to infer features which are specific to each class. Each reconstructed sample looks like an averaged record over a class and can be considered as the representative of it. Similarly, an adversary who participates actively in the learning process as it is the case in federated learning can use GAN [Goo+14] to reconstruct the representatives of the classes [HAP17]. And in the special case where all the records belonging to the same class represent the same person or thing then both inversion attack and GAN attack can reconstruct samples which are similar to training inputs. Recent reconstruction attacks [ZLH19; ZMB20; GBDM20] show that the complete training samples including their labels can also be reconstructed only from the captured gradients. However, the reconstruction accuracy is inversely proportional to the batch size which is used to compute those gradients. Indeed, the received gradients on which the attack is based is the average gradients over a batch and more we increase the size of the batch, more we dilute the effect of any individual record, thus becoming harder to reconstruct it.

In [NSH19], the proposed attack infers if a specific person is included in the training dataset of the participants (aka, Membership inference). The adversary extracts the following features from every snapshot of the common model, which is a neural network: output value, hidden layers, loss values, and the gradient of the loss with respect to the parameters of each layer. These features are used to train a membership inference model, which is a convolutional neural network. The attack is more powerful when the adversary is the server and therefore has access to each individual update and when it is active.

3.1.2 Privacy-Preserving Machine Learning

Differentially Private machine learning models are obtained by adding a calibrated amount of noise during the learning process. More, specifically the noise is added on: (1) the objective function (objective perturbation) [CMS11; JT14; PWWD16; PWD17; Iye+19; JKT12; CM09; ZZXYW12], (2) the gradients at each iteration (gradient perturbation) [SS15; ZWZZC19; SCS13], (3) the final model obtained after the training (output perturbation) [CMS11; JT14; CM09; PRR10; HCB16; Wu+17], (4) input perturbation [DJW13] and (5) label perturbation via teachers-student ensemble [PAEGT16]. However, the output and objective perturbation methods are often not applicable to non-convex settings as it is the case in deep learning. Generally, gradient perturbation is widely used for differentially private deep learning [SS15; ZWZZC19; Aba+16] and requires to manually clipping the gradients norm at each iteration as we are not able to have a prior estimate of the real bound in deep learning. Clipping the norm and therefore bounding the gradients' sensitivity is required to generate the noise in differential privacy [Aba+16; JE19].

The concept of Client-based Differential Privacy has been introduced in [MRTZ18] and [GKN17], where the goal is to hide any information that is specific to a single client's training data. These algorithms noise the contribution of a single client instead of a single record in the client's dataset. The noise is drawn from continuous distributions and added by the server, hence, unlike our solution, these works assume that the server is trusted. However, a noise drawn from continuous distributions may impact negatively the differential privacy guarantees [Mir12]. In fact, the vulnerability is based on irregularities of floating-point implementations of the privacy-preserving mechanism. Therefore, a better solution would be to sample noise from discrete distributions instead [Mir12; CKS20]. The use of Discrete Gaussian has been proposed in [CKS20], however it requires to add the discrete noise on integer values. Fortunately, our discrete Gaussian mechanism proposed in Section 5.2.2 is more general as the noise can be added even to real numbers. Moreover, the privacy guarantee of the distributed generation of discrete Gaussian noise has not been analyzed earlier (e.g., in federated learning).

Recently, [LCYC20] proposed to add noise only to the K largest update [1] à la local-DP. In local-DP, each client adds larger noise that what is necessary to guarantee DP for the *aggregated* model update without using secure aggregation. Therefore, the common model is less accurate than with our scheme (aka. FL-TOP-DP and defined in Section 5.4.1). In addition, [LCYC20] uses two epsilon budgets; one for selecting Top-K parameters per client, and the second for perturbing these selected Top-K parameters. By contrast, in FL-TOP-DP, we select the Top-K parameters via public data without sacrificing any privacy budget. Finally, their solution is also less bandwidth efficient than ours: as the Top-K parameters differ for each client and at each round, the client cannot send only the Top-K parameters values because the server will not be able to identify which value corresponds to which Top-K parameter. For this reason, the client has to send a sparse vector with only Top-K perturbed values and all remaining parameters set to 0. Therefore, the quantization of the non-Top-K parameters is performed only during the upstream (from client to server) without compressing any downstream traffic. As opposed to this, in our solution, only the weights/updates of the Top-K parameters are transferred downstream/upstream.

In [JHHDW20], a private extension of SignSGD protocol defined in [BWAA18] was proposed. The authors used local DP to guarantee client-level-DP. Their proposal is more bandwidth efficient than our private protocol called FL-SIGN-DP and presented in Section 5.2 as they send only one bit per parameter, however, it is widely accepted that the large perturbation provided by local DP degrades accuracy: the aggregation of the DP updates increases the noise variance. Our private proposals add noise in a distributed manner such that the final noise after the aggregation corresponds to the minimum noise needed to ensure DP. This is made thanks to a secure aggregation protocol. Regarding the Byzantine resilience of their protocol, only the attack where each adversary sends the opposite sign of its gradients is considered while we consider the gradient ascent and the random update attacks. Furthermore, we consider backdoor attacks, and more powerful adversaries which collude by sharing their data and agree on the same poisoning updates.

[1] We refer to the model parameters of the K most largest updates as Top-K parameters.

In [NHC21], the authors show how DP variants (Local or Central DP) defend against backdoor attacks. However, the adversaries in this approach either faithfully follows the DP protocol and do not deviate from it (under Local DP settings) or in case of Central DP the server is trusted and it takes care of adding the noise. In our case, we make the opposite observations if the adversaries decide to send their non-noisy, boosted and polluted updates.

Compressive Sensing (CS) [Don06; CT06; CRT06; JV10] was used with DP in [LZWY11]. The authors show that the amount of noise is reduced from $O(\sqrt{n})$ to $O(\log(n))$, when the noise is added on the sampled coefficients instead of the original database.

Cronus uses adversarial regularization techniques [NSH18] to be more robust against active and passive membership attacks. However, using local DP to have a theoretical privacy-preserving guarantee degrades the model accuracy of the global model such that each single participant's model (without collaboration) is more accurate.

The authors in [Mo+21] proposed to use hardware-based Trusted Execution Environments (TEEs) in Federated Learning settings. Indeed, TEEs allow to securely store data and execute code on an untrusted device. This proposed approach is an alternative to solutions which incur significant computational overhead as it is the case when we use multi-party computation or fully homomorphic encryption [NLV11; 20]. However, even if the results are promising it still introduces increase in 15% CPU time, 18% memory usage and 21% energy consumption overhead in the client-side. Moreover, creating large TEEs is considered to be a bad practice as it has proven to significantly increase the attack surface. Therefore, the TEE size must be as small as possible. As a consequence, deep neural networks, which have generally large size, cannot be trained. As a solution, the author proposed to train each layer or block of a layer one after the other based via greedy layer-wise training [BLPL07; LBLL09]. In [YBS20] authors decided to investigate the utility of being a participant during the federated learning process. As a result, the authors have shown that some participants have local models which are more accurate on their data than the global model. Moreover, the use of robust aggregation or differential privacy increases this gap. Finally, they evaluate three well-known techniques: fine-tuning, multi-task learning, and knowledge distillation for local adaptation of federated models. Therefore, the local adapation of federated models yields to outperform all local models.

3.2 Bandwidth Efficient Machine Learning

Different quantization methods have been proposed to save the bandwidth and reduce the communication costs in federated learning. They can be divided into two main groups: unbiased and biased methods. The unbiased approximation techniques use probabilistic quantization schemes to compress the stochastic gradient and attempt to approximate the true gradient value as much as possible [AGLTV17; Wen+17; Wan+18; Kon+16]. However, biased approximations of the stochastic gradient can still guarantee convergence both in theory and practice [BWAA18; LHMWD18; SFDLY14]. SignSGD [BWAA18] protocol uses quantization to reduce the number of required bits per weight/update value during downstream and upstream exchanges between the server and the clients but requires the use of all the clients at each round which is not realistic in the context of federated settings because each client is available only during few rounds [Kai+19].

A different line of works exploit the sparsity of model updates to compress model updates. Our work presented in Section 4.1.4 belongs to this line. The authors in [MKAV11] use CS for low-complexity energy-efficient ECG compression. Although compressed sensing was primarily designed for compression [CRT06; Don06], it can be used for the purpose of denoising [MMB16; TP12]. In [YZT14], compressed sensing based denoising and certain artificial intelligence are combined to improve the prediction performance.

Similarly to our solutions presented in Section 4.1.4, authors in [AG19; AG20] proposed to use compressive sensing for federated learning in order to compress model updates. However, their solutions are non-private and assume that all clients participate in each round (as they maintain an error accumulation vector at each client due to the compression scheme), but as discussed in [Kai+19] this assumption is not always realistic. Recently in [JALP21] another compressive sensing algorithm

was proposed for denoising purpose in federated learning context, where the added noise is due to the network transmission.

Sketching was adapted to federated learning for the purpose of compressing model updates in [IRUSA+19]. The authors proposed to use Count-Sketch [CCF02] to retrieve the largest weights in the update vector on the server side. After that, the server uses two additional communication rounds to inform the clients about what gradient values they need to send back to the server. The server then takes the average of the received gradients and zeros-out the others before updating the model. The error due to the compression is maintained at each client, and the participation of all clients are required in each round which, as per [Kai+19] and as discussed above, is not practical to federated learning. In [Rot+20], the aforementioned scheme is improved further by directly retrieving the most updated gradient values without asking for their positions in the update vector. This makes the scheme more efficient as it needs fewer communication rounds. Similarly to our approach called FL-CS and presented in Section 4.1.4, the error vector is also maintained on the server side instead of the client side, which is clearly a better fit for federated learning.

Constraining the weights to have a specific distribution has already been studied. In [Han+16], for example, the authors use pruning techniques to create a sparse model at the end of the training. After each SGD iteration, the authors zero-out all the weights with an absolute value smaller than a threshold. Iterating the process leads to a sparse model with only some absolute weight values larger than 0. Similarly, [CHSEB16] aim to create a model with binary weights such that at the end of the training all the weights are close to 1 or -1. After each SGD update, the authors take the sign of the weights before the next update. After some iterations, the weight values become close to the interval limits -1 and 1.

In [FC18], a new hypothesis claims that we can extract a sub-neural networks from a model which, if trained, can achieve similar performance. To find such a sub-network, one has to follow a simple iterative procedure: train the complete network, prune the smallest weights, and then reinitialize the remaining weights to their original values. These steps are repeated iteratively. This approach was extended to federated learning in [Li+20].

3.3 Machine Learning and Security

In this section, we focus on the poisoning attacks against Federated learning as it is the most studied attacks in federated context. We also investigate the potential defenses against such attacks in literature.

3.3.1 Poisoning Attacks

During the training, the poisoning attacks can be performed on the model (model poisoning) [BEGS17; BZAA18] or on the data (data poisoning) [BNL12; Rub+09; MZ15; Xia+15; KL17; CLLLS17; Jag+18; STS16; FYB18]. In what follows, we will first describe the difference between Model Poisoning and Data Poisoning and therefore we explain the goal behind such attacks (targeted/untargeted).

Model Poisoning vs Data Poisoning

With data poisoning attacks only the data are modified by the adversary while the learning process remains unchanged [TTGL20]. By contrast, the adversary in model poisoning attacks, modifies its learning process instead of its data to generate adversarial updates. However, both can be combined to reach the adversarial objective [BCMC19; BVHES18]. Although both are effective, data poisoning is much more practical than model poisoning and it requires much less expertise from the adversary [TTGL20].

Data poisoning attacks largely fall in two categories: clean-label[Sha+18; Muñ+17; KL17] and dirty-label [GDG17; CLLLS17; Liu+18]. The first one assumes that the adversary cannot change the label of any training data due to a process which certifies that data are belonging to the correct class and the poisoning of data samples has to be imperceptible. In contrast, with dirty-label attacks the adversary can modify the labels of some training data samples to miss-classify them to the desired targeted label in order to generalize this behavior on the testing data. Dirty-labels attacks includes

		Attack Approaches			
		Targeted Poisoning	Untargeted Poisoning	Model Poisoning	Data Poisoning
Attacks	Random Update	✗	✓	✓	✗
	Ascent Gradient	✗	✓	✓	✗
	In-Backdoor	✓	✗	✗	✓
	Out-Backdoor	✓	✗	✗	✓

Tab. 3.1: Summary of the attacks which are considered in our Security analysis introduced in Chapter 6. The description of all the attacks can be found there.

backdoor attacks which are performed during the training phase either by label-flipping attack which modifies the label of a specific class to targeted class's label, or by modifying specific features of some samples to insert a timestamp or a trigger before to change their labels to desired class's label as with label-flipping. After that if the trigger is available on some testing data then the model will behave according to the adversary's target by predicting the targeted class. Note, that in comparison with the phenomenon of adversarial examples [CW17; Sze+13] which aims to modify only the inputs of the testing data to carry out a miss-classification, with dirty-label we aim at modifying the testing data according to the training data. So, we insert for example a trigger in the testing data only because it was present in the training data.

Targeted vs Untargeted poisoning attacks

The poisoning attack can be either targeted where the goal of the adversary is to cause the misclassification of specific class in the global model while maintaining a good accuracy on the non-targeted classes (Backdoors attacks [BCMC19; BVHES18]); or untargeted where it aims to decrease the accuracy of the global model without distinction between the classes [HJNRT11; LYY20; BEGS17]. The first backdoor attack designed for a federated learning environment was proposed in [BVHES18]. Here, the adversary scales up its update in order to surpass the contributions of other honest participants after aggregation. The goal of the attack is to alter the common model so that it exhibits some adversarial behaviour (e.g., targeted misclassification). However, these attacks are effective only in later rounds, when the global model has converged. Indeed, the attack exploits the fact that when the global model has converged, the updates of other honest clients will be smaller and then are more easier to surpass. In contrast, the adversary in [BCMC19] boosts its update enough to surpass the contributions of honest clients from the very first rounds even when the global model has not converged. The papers [BCMC19; BVHES18] show how to bypass detection and carry out a stealthy model poisoning. Indeed, in [BVHES18] the adversary adds a specific term to the loss function such that it can optimize the model for the backdoor task while it remains close enough from the non-backdoor model to avoid detection. In [BCMC19], an additional term is added to ensure the minimum degradation on the accuracy of the global model in order to evade accuracy-based detection. Moreover, the authors alternate between the minimization of the stealth and the adversarial loss which separates one from the other for a finer control over relative effect of the two objectives. The authors claim that the alternating minimization strategy increases the targeted objective while it increases the ability of the attacker to evade detection.

The resilience of distributed implementations of Stochastic Gradient Descent (SGD) against Byzantine failures is studied in [BEGS17]. Each Byzantine worker (among a set of workers) sends a random vector drawn from a Gaussian distribution. The results show that only a single Byzantine worker can prevent the traditional federated schemes such as FL-STD from converging. Note that this attack can be adapted to SignSGD scheme, where an adversary sends the sign of the random value instead [BZAA18].

Although label flipping is often used as targeted class to insert a backdoor [BCMC19; TTGL20], however, it can also be used as untargeted attack if the adversary decide to flip all the labels of her local training data randomly and without distinction between classes [Jag+18; Muñ+17].

In Table 3.1 we classify the studied poisoning attacks used in our Security Analysis in Chapter 6.1.

3.3.2 Defenses

KRUM a Byzantine-resilient algorithm is proposed in [BEGS17] as an aggregation rule to select one honest update per round in an adversarial environment. In [YCKB18] two distributed gradient descent algorithms were proposed: the first one is based on the simple coordinate-wise median operation

while the second is based on the coordinate-wise trimmed mean operation where for each coordinate, only participants with value smaller or larger than a constant are taken into account to compute the mean value. The original version of the trimmed mean operation for collaborative learning was introduced in [EGR18] where the author introduced a hybrid aggregation rule called Bulyan which combines both KRUM and the trimmed mean operations to be more efficient. Another version of the trimmed mean is presented in [XKG18]. In [FYB18], a solution was proposed to defend against backdoors in federated learning. Same, paper [TTGL20] proposes a clustering-based solution to detect malicious participants and then to defend against backdoor attacks (Label Flipping). Indeed, the insight behind this solution is that the updates of malicious and honest participants are distinguishable. Therefore, by using PCA clustering on the weights of the last layer's nodes, the server can separate the two groups of participants. Neural Cleanse [Wan+19] is a technique which aims at detecting and removing backdoors in deep neural networks. This technique assumes that the clusters of the source and target classes are close in the representation space under backdoor attacks. [STS16] demonstrates how to circumvent backdoor attacks on distributed learning by using a variant of Trimmed Mean which is based on K-mean clustering. In [BZAA18], the authors study the robustness and the tolerance of signSGD/SIGNUM [BWAA18] with majority vote against network faults and adversarial clients, where SIGNUM is the momentum equivalent of signSGD (i.e., each client maintains a momentum and transmits the sign momentum to the server at each iteration). In [BZAA18], the authors show that signSGD is robust against sign inversion attack, when each malicious client inverts the sign of the computed gradient. The authors argue that this is the best possible attack in a *non-adaptive* setting (i.e., when the adversary performs the attack independently of the gradients it computed). In Section 6, we experimentally show that FL-SIGN is also robust against other adaptive attacks like various backdoor attacks [BCMC19]. Multiplicative weight update (MWU) technique [AHK12; FS97; PST95; Li+14; GK07] defines a set of weighted aggregation rules which are robust against poisoning attacks. The intuition behind MWU-based aggregations is to reduce the impact of malicious participants on the weighted aggregated update by reducing their weights. Finally, Cronus [CSSH19] is a robust defense scheme against poisoning attacks. The scheme use knowledge transfer through distillation between the server and different parties. This implies that the only information shared between them is their prediction on a public dataset. Moreover, the server replace the naive aggregation rules commonly used in federated learning scheme (mean, averaged mean) with the robust mean estimation proposed in [Dia+17]. Cronus outperforms most of the aforementioned defense schemes such as Median [YCKB18], Bulyan [EGR18], Krum [BEGS17], MWU with mean aggregation and MWU with optimization [Li+14].

How effective are they?

Although different defenses were proposed to detect or cancel-out such attacks, however, recent works show that an adversary is able to bypass and evade numerous of them. Indeed, [BBG19] shows that by carefully crafting the byzantine values, an adversary which controls some participants can defeat most of the state-of-the-art defenses including: Krum, Trimmed Mean and Bulyan. Two attacks was proposed: the first one aims to prevent the convergence while the second aims to backdoor the model. Moreover, the paper clearly shows that the assumption which is made by most existing defenses for distributed learning [BEGS17; EGR18; XKG18; YCKB18] that if each value from the adversary update is upper-bounded by the variance of the honest workers' values then the attack fails, is not correct. Indeed, even by choosing those values from that range, it is possible to succeed in the attack. In fact, those defenses use statistical methods to remove all the updates with large changes to prevent the attacks. Similarly, the solution proposed in [STS16] to circumvent backdoor attacks on distributed learning was successfully evades it by the backdoor attack introduced in [BBG19]. Note also, that the KRUM and the coordinate-wise median protocols are not effective against the backdoor attack used in [BCMC19].

Finally, except for signSGD, all the other defenses aforementioned assume to have access to each individual update in order to remove measure and perform the statistics-based methods to remove potential adversaries. Therefore, secure aggregation can not be used as it allows to access only to the aggregation while each individual update is encrypted.

Part II

Contributions

4 Toward Bandwidth Efficient Federated Learning Schemes

Federated Learning is bandwidth hungry and not adapted to applications that use limited bandwidth devices, such as IoT devices. Indeed, it is generally used to train deep neural networks with millions of parameters and each parameter is encoded on 32 bits. Also, it requires more than one round to converge. Moreover, the communication exchanges between the server and the clients in both directions (upstream and downstream), increase drastically the energy consumption. Therefore, Federated Learning is not practical for applications which are based on energy-constrained devices.

To make it more practical, it is useful to study and propose some federated schemes which are adapted to energy and bandwidth-constrained devices.

To address this issue, we studied two approaches namely update compression and model compression, and proposed three bandwidth efficient schemes:

- FL-SIGN is a scheme based on biased quantization of the updates and enables to reduce the number of bits per weight's update from 32 bits to 1 bit.
- FL-CS for its part relies on the compressive sensing theory which enables to reconstruct a complete sparse data (ie,. image or signal) from few sampled points in it. Analogously, starting by some points sampled from the update, FL-CS leverages its sparsity in order to reconstruct it accurately. Therefore, the model's update is compressed by up to 95%.
- FL-TOP harnesses the ability of the model to converge and reach good accuracy under constraint. Indeed, in FL-TOP only some weights are updated while keeping all the remaining ones constant. As a result, only the few updated weights have to be exchanged between the server and the clients. FL-TOP enables to reduce the model size up to 99.9% in both downstream and upstream directions which makes it particularly adapted to applications based on energy-constrained devices as it is the case for mobile systems.

All the proposed solutions obtained similar accuracy to the standard Federated Learning scheme in spite of their bandwidth efficiency.

4.1 Reducing Bandwidth by Compressing the updates

Update compression consists in compressing the size of the updates that are sent by the participants to the server. We proposed two "update compression" solutions, namely FL-SIGN and FL-CS.

4.1.1 FL-SIGN: Bandwidth-Efficient Federated Learning via Quantization

The FL-SIGN Protocol

Algorithm 3: FL-SIGN: Sign Federated Learning

Server:
  Initialize common model w_0
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\mathbf{s}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \gamma \mathsf{sign}\left(\sum_k \mathbf{s}_t^k\right)$
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\mathbf{w}_{t-1}^k)$**:**
  $\mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
  Output: Model update $\mathsf{sign}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k)$

We propose to reduce the bandwidth costs by quantizing the model weights as in [BZAA18]. More specifically, in the new scheme, referred to as FL-SIGN in the rest of this dissertation, each client sends only the sign of every coordinate value in its parameter update vector. The server takes the sign of the sum of signs per coordinate and scales down the result with a fixed constant γ (which is in the order of 10^{-3} in practice) in order to limit the contribution of each client and adjust convergence. This scaled aggregated updates are added to the global model.

More specifically, FL-SIGN (see Alg. 3) differs from the standard federated scheme FL-STD (see Alg. 1) as follows:

1. Each client returns $\mathbf{s}_t^k = \mathsf{sign}(\mathbf{w} - \mathbf{w}_{t-1}^k)$ instead of $(\mathbf{w} - \mathbf{w}_{t-1}^k)$, where $\mathsf{sign} : \mathbb{R}^n \to \{-1, 1\}^n$ returns the sign of each coordinate value of the input vector if it is non-zero and a sign chosen uniformly at random otherwise.

2. The server sums the sign vectors $\mathbf{s}_t^k$ sent by each client k and computes the sign vector of this sum as $\mathsf{sign}\left(\sum_k \mathbf{s}_t^k\right)$. This is equivalent to take the median of all clients' signs at every position of the update vectors. Unlike in Alg. 1, the update $\mathbf{s}_t^k$ is *not* weighted with client k's data size $|D_k|$, since that would require the client to send $|D_k|$ to the server which would enable the adversary to maliciously scale up its sign vector by sending a fabricated size of its training data.

The extreme quantization performed by FL-SIGN reduces the communication costs of federated learning by a factor of 32 (since only one bit is sent per parameter instead of 32 bits), and also, as we will demonstrate later (in Chapter 6), improves its robustness against different attacks aiming to maliciously manipulate the common model through the updates. Note also that, if the quantized update vector is sparse, other compression techniques can further improve communication efficiency [Kon+16].

In signSGD [BWAA18], all the clients calculate the stochastic gradient based on a single mini-batch and then send the sign vector of this gradient to the server. The server calculates the aggregated sign vector by taking the median (majority vote) and sends the signs of the aggregated signs back to each client.

The main differences between our scheme (FL-SIGN) and signSGD are as follows:

- FL-SIGN aims to train a common model that is distributed to a random subset of all clients in every round. However, in signSGD, all clients start with the same initialized common model and the server sends the same aggregated model update to *every* client at each round. Selecting

only a random subset of clients in each round has at least three benefits. First, FL-SIGN becomes more robust against temporary node failures. Second, FL-SIGN reduces the communication costs upstream to the server. Finally, sampling boosts privacy due to the uncertainty that a specific user's or client's data is used for training or not.

- In FL-SIGN, each client can perform multiple SGD iterations locally using multiple mini-batches before computing the model update. On the contrary, signSGD always performs one local SGD iteration with a single mini-batch at every client.
- As all the clients participate at each round in signSGD, the server only transfers the sign of the aggregated signs to the clients in every round. Therefore, only a single bit is transferred per parameter downstream to the clients. In FL-SIGN, the whole model is transferred but only to a random subset of clients.

4.1.2 Experimental Set-up

This section describes the experimental set-up that are used to evaluate the accuracy, security and privacy of our proposals in the rest of the paper. The following datasets were used: MNIST, Fashion-MNIST, IMDB, LFW and CIFAR which is augmented from 50,000 images to 500,000 (See Appendix 8.3 for more details)

The description of the datasets and model architectures can be found in Section 8.3 of the Appendix. For FL-SIGN, γ, the learning rate, was set to 0.001 for all datasets[1]. N, the total number of participant clients, was set to 1000. C, the percentage of selected clients at each round, was set to 0.1. $|D_k|$ is the training data size of client k. $|\mathbb{B}|$, the batch size, was set to 50 with CIFAR dataset, 25 with IMDB, 10 for MNIST and Fashion-MNIST datasets. T_{gd}, the local gradient descent iterations per round and per client, was set to 30, 30, 5 and 50 for MNIST, Fashion-MNIST, IMDB and CIFAR, respectively. T_{cl}, the number of rounds, was set to 100 for the MNIST, Fashion-MNIST, IMDB datasets, and 400 for CIFAR. We use two optimizers: the stochastic gradient descent (SGD) [Cho+15d] with a learning rate (η) set to 0.215 and the adaptive moment estimation (Adam) [KB14] [Cho+15d] with a learning rate set to 0.001. Table 8.1 summarizes the different parameter values that were used for the different datasets.

4.1.3 Performance Evaluation

In this section, we compare the performance of FL-SIGN and FL-STD using the same configuration. The global model accuracy of FL-STD and FL-SIGN on the CIFAR, MNIST, Fashion-MNIST, IMDB datasets are compared in Table 4.1. The bandwidth consumption is calculated by measuring the average number of bits sent by a client to the server. This is computed as ($C\times$ round $\times\ n$) for FL-SIGN, and ($32 \times C\times$ round $\times\ n$) for FL-STD, where n is the model's size and round represents the round when we get the best accuracy over T_{cl} rounds. Similarly, we present the best accuracy over T_{cl} rounds.

The results show that the accuracy performance of both schemes over the five datasets are very similar despite the severe parameter quantization. Indeed, the difference between FL-STD and FL-SIGN in terms of accuracy is between 0.01 and 0.03. However, FL-SIGN is much more bandwidth efficient and consumes up to 59 times less bandwidth than FL-STD.

Dataset	FL-STD			FL-SIGN		
	Acc	round	Cost (Megabytes)	Acc	round	Cost (Megabytes)
CIFAR	0.86	375	205.46	0.83	386	6.61
MNIST	0.99	88	58.55	0.98	48	1.0
Fashion-MNIST	0.89	90	59.88	0.87	68	1.41
IMDB	0.88	84	13.53	0.85	91	0.46

Tab. 4.1: Summary of results using the FL-SIGN scheme.

4.1.4 FL-CS: Bandwidth-Efficient Federated Learning via Compressive Sensing

In this section, we define our compressive sensing-based bandwidth-efficient scheme called FL-CS.

The FL-CS Protocol

[1] We noticed experimentally that γ should be selected between range 0.001 and 0.005. And it should be increased when DP is used.

Algorithm 4: FL-CS: Federated Learning

1 **Server:**
2 Initialize common model $\mathbf{w}_0$, η_G , ρ, $\mathbf{u}_t = 0$, $\mathbf{e}_t = 0$
3 **for** $t = 1$ **to** T_{cl} **do**
4 Select $\mathbb{K}$ clients uniformly at random
5 **for** *each client* k *in* $\mathbb{K}$ **do**
6 $\mathbf{y}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1})$
7 **end**
8 $\mathbf{y}_t = \sum_{k=1}^{|\mathbb{K}|} \frac{\mathbf{y}_t^k}{|\mathbb{K}|}$: Averaging
9 $\mathbf{u}_t = \rho \mathbf{u}_{t-1} + \mathbf{y}_t$: Momentum
10 $\mathbf{e}_t = \eta_G \mathbf{u}_t + \mathbf{e}_{t-1}$: Error Feedback
11 $\mathbf{s}_t = \mathcal{D}(\mathbf{e}_t, n)$: Reconstruction
12 $\mathbf{e}_t = \mathbf{e}_t - \mathcal{C}(\mathbf{s}_t, m)$: Error accumulation
13 $\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{s}_t$: Update
14 **end**
Output: Global model $\mathbf{w}_t$
15
16 $\mathbf{Client}_k(\mathbf{w}_{t-1}^k)$**:**
17 $\mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
18 $\Delta \mathbf{w}_t^k = \mathbf{w}_t^k - \mathbf{w}_{t-1}^k$
Output: Model update $\mathcal{C}(\Delta \mathbf{w}_t^k, m)$

CS assumes the sparsity of the reconstructed signal in a specific basis domain Ψ as explained in Section 2.3. We assume the model update (as a signal) to be already sparse in the time domain, that is, Ψ is canonical sparsity basis (i.e., $\Psi = \mathbf{I}$), and therefore, the compression operator is $\mathcal{C}(\Delta \mathbf{w}, m) = \Phi \Delta \mathbf{w}$, where Φ is composed of the first m rows of the matrix of the Discrete Cosine Transform (DCT) [ANR74; Ahm91]. Indeed, due to the large energy compaction property of DCT, the first coefficients, which correspond to the low frequency components of $\Delta \mathbf{w}$, tend to have the largest magnitude and hence convey the most information about the model update [RY14]. In fact, for a canonical sparsity basis $\Psi = I$, $\Theta = \Phi$ is RIP with overwhelming probability as soon as $m = O(U \ln^4 n)$ if $\Delta \mathbf{w}$ is U-sparse [CT06]. Therefore, reconstruction is possible according to Theorem 1.

The decompression operator $\mathcal{D}$ is defined Eq. (2.4). Note that the compression operator can be computed in $O(n \ln n)$ with FFT and the decompression (or reconstruction) operator is implemented with the OWL-QN algorithm [AG07] which makes our approach reasonably fast in practice.

FL-CS is described in Alg. 4. A client first computes its update $\Delta \mathbf{w}_t^k$ with SGD, and then transfers the compressed update $\mathcal{C}(\Delta \mathbf{w}_t^k, m)$, which consists of the first m DCT coefficients of the update (Line 18). The server takes the average of the client's updates (Line 8), updates the momentum (Line 9), and computes the error $\mathbf{e}_t$ (Line 10-12) due to compression following the error propagation technique described in Section 2.3.1. This error is accumulated over all federated rounds and added to the model (Line 13) to compensate its negative effect on convergence. The server uses OWL-QN [AG07; Tay20] to reconstruct the error-compensated aggregated model update $\mathbf{s}_t \in \mathbb{R}^n$. Finally, the server updates the global model as $\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{s}_t$ before re-distributing the updated model to a new set of clients $\mathbb{K}$.

Notice that the error $\mathbf{e}_t$, the averaged model update $\mathbf{y}_t$, as well as the momentum are maintained in the compressed domain and have a size of m instead of n. Moreover, all of them are maintained at the server side instead of at each client side. This is possible due to the linearity of the compression scheme which is detailed in Section 2.3.

Scalable Reconstruction: Although OWL-QN is reasonably fast in practice, its computational overhead may not be tolerated with very large models. A scalable reconstruction is proposed as follows. On the client side, the update vector $\Delta \mathbf{w}_t$ is shuffled and then splitted into P equally-sized chunks. Then, the compression operator $\mathcal{C}$ is applied on each individual chunk. Finally, the compressed chunks are transferred to the server. On the server side, each chunk is reconstructed independently using OWL-QN. The decompressed chunks are concatenated, and the resulted vector with size n is reshuffled to obtain $\mathbf{s}_t$ by inverting the client-side shuffling.

Shuffling is performed by each client identically which guarantees that the compressed chunks can still be aggregated by the server. In practice, this can be implemented by sharing a common random

seed among all participants to initialize the shuffler. As the server also knows this seed, it can invert this shuffling and reconstruct the aggregated model updates. Note that shuffling is also performed identically over all the rounds to maintain the error.

Notice that, instead of reconstructing the complete update vector at once, the server performs reconstruction on smaller chunks which makes decompression faster. In addition, shuffling guarantees that the sparsity of the chunks is proportional to the sparsity of the whole update vector (i.e., if the update vector is U-sparse then all its chunks are U/P-sparse). Hence, the same compression operator $\mathcal{C}(\cdot, m/P)$ can be applied on every chunk without increasing the compression ratio (i.e., the compressed update still has a size of m). Moreover, each chunk can be reconstructed independently in parallel which can significantly speed up decompression.

4.1.5 Experimental Set-up

This section describes the experimental set-up that are used to evaluate the accuracy of our proposal. We used the Fashion-MNIST and the Medical dataset and their respective models described in Section 8.3.

For both datasets, we tune η from 0.01 to 0.5 with an increment value of 0.005. As in [IRUSA+19], we fix the momentum parameter ρ to 0.9 and we tuned the global learning rate η_G from 0.05 to 2.0 with an increment value of 0.05. The number of chunks used is $P = 200$. The hyperparameters used by each of the considered schemes are summarized in Table 8.4 and Table 8.5 in the Appendix for Fashion-MNIST and the medical dataset, respectively.

We aim at evaluating the performance of FL-CS with different levels of compression and comparing them with the performance of the following learning protocols:

- FL-STD: It is described in Section 2.1 (see Alg. 1).
- SignSGD: It is descibed in Section 4.1.1.
- FL-RND: This baseline follows the algorithm of FL-STD except that a random subset of the update vector with size $m \leq n$ is sent to the server instead of the complete update of size n. Each client selects the same random subset of coordinates from the update vector, but a different subset in every round. The server then averages the received updates before updating only the corresponding m weights. Note that if $m = n$, FL-RND is equivalent to FL-STD (see Alg. 5).
- FL-FREQ: In this baseline, a client transforms the model update to the frequency domain by using DCT [ANR74; Ahm91], and then the first m coefficients (low frequency components) are extracted and sent to the server as in FL-CS. However, as opposed to FL-CS, the server reconstructs the aggregated update vector by applying the inverse DCT on the aggregated compressed vectors where the last $n - m$ coefficients are zeroed out (see Alg. 6). This baseline corresponds to a low-pass filter applied on the update vector. Φ in Alg. 6 is composed of the first m rows of the matrix of the DCT.

4.1.6 Performance Evaluation

Table 4.2 represents the best accuracy over 200 rounds for each scheme on the Fashion-Mnist dataset. $Round$ corresponds to the round when the best accuracy is reached and $Cost$ is the average bandwidth consumption calculated as: $r \times n \times 32 \times Round \times C$, where 32 is the number of bits necessary to represent a float value, n is the uncompressed model size, $r = \frac{m}{n}$, m is the compressed model size, C is the sampling probability of a client, and $Round$ is the round when we get the the best accuracy.

Table 4.3 represents the best balanced accuracy over 100 rounds for each scheme on the Medical dataset. AUROC (area under the receiver operating characteristic curve [Nar18]) corresponds to the AUROC value when the best balanced accuracy is reached, round is also the round when we get the best balanced accuracy, and finally, Cost is the average bandwidth consumption calculated as for the Fashion-MNIST dataset described above.

Algorithm 5: FL-RND

Server:
  Initialize common model $\mathbf{w}_0$
  for $t = 1$ **to** T_{cl} **do**
    Generate a random seed ζ Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\mathbf{y}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1}, \zeta)$
    end
    $\mathbf{y}_t = \sum_k \frac{|D_k|}{\sum_j^N |D_j|} \Delta \mathbf{w}_t^k$
    $j = 0$
    for *each element* i *in* $\mathbf{G}$ **do**
      $\mathbf{w}_t[i] = \mathbf{w}_{t-1}[i] + \mathbf{y}_t[j]$
      $j = j + 1$
    end
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\mathbf{w}_{t-1}^k, \zeta)$**:**
  $\mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
  $\Delta \mathbf{w}_t^k = \mathbf{w}_t^k - \mathbf{w}_{t-1}^k$
  Generates a random set $\mathbf{G} = \{x \in \{1, \cdots, n\}\}$ of $\mathbf{m}$ random integer values such that $\mathbf{m} \leq \mathbf{n}$ based on the seed ζ
  $\hat{\Delta w_t^k}$ =Sample $\mathbf{m}$ elements from $\Delta \mathbf{w}_t^k$ by taking each element of $\mathbf{G}$ as a coordinate
  Output: The sampled Model update $\hat{\Delta w_t^k}$

Without DP, notice that our FL-CS scheme outperforms FL-RND and FL-FREQ whatever the considered compression ratio or the dataset are. Also, compared to FL-STD, our scheme started to reach the same accuracy from a compression ratio r being equal or greater than 0.1 for both datasets, although the differences between FL-CS and FL-STD for a compression ratio of 0.05 are only of 6% [2] and 1% [3] for the Fashion-Mnist and the medical datasets, respectively. However, FL-STD consumes much more bandwidth than FL-CS. Indeed, FL-CS reduces the bandwidth cost by 95% compared to FL-STD with a compression ratio of 0.05 for both datasets, while the bandwidth cost is reduced to 80% and 85% with a compression ratio of 0.2 for Fashion-MNIST and the medical data, respectively.

SignSGD for its part, also reaches similar accuracy than FL-STD. The bandwidth cost, SignSGD consumes less bandwidth than FL-STD with the Medical dataset as it reaches the best accuracy earlier than FL-STD. However, for the Fashion-MNIST dataset, FL-STD is more bandwidth efficient than SignSGD. It can be explained by the fact that only a small proportion of available clients are selected at each round in FL-STD, while in SignSGD all the clients are selected at each round. Notice, that our bandwidth efficient schemes outperform SignSGD in term of bandwidth efficiency irrespective of the considered dataset.

4.2 Reducing Bandwidth by Compressing the Model

Model compression consists in compressing the size of the model in order to reduce the number of element in the update vectors. We developed a scheme, called FL-TOP, that harnesses the ability of the model to converge and reach good accuracy under constraint.

4.2.1 FL-TOP: Bandwidth-Efficient Federated Learning via Constraint

In the standard federated learning scheme (FL-STD, in Section 2.1), the server sends the latest updated model to a randomly selected set of clients (downstream), and each client sends back its complete model update after local training to the server (upstream) at each round. Knowing that a model has on average millions of parameters (each is a floating point value represented on 32 bits), the network can suffer from large traffic both upstream and downstream.

Our solution, called FL-TOP, aims to reduce the large amount of network traffic by reducing both downstream and upstream traffic. In what follows, we describe the non-private scheme FL-TOP.

[2] Based on the accuracy
[3] Based on the balanced accuracy [BOSB10; BDA13]

Algorithm 6: FL-FREQ

Server:
  Initialize common model $\mathbf{w}_0$
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\Delta \mathbf{y}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{y}_t = \sum_k \frac{|D_k|}{\sum_j^N |D_j|} \Delta \mathbf{w}_t^k$
    $\hat{\mathbf{y}}_t = \Phi^{-1}\mathbf{y}_t$: Transform to time domain
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \hat{\mathbf{y}}_t$
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\mathbf{w}_{t-1}^k)$**:**
  $\mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
  $\Delta \mathbf{w}_t^k = \mathbf{w}_t^k - \mathbf{w}_{t-1}^k$
  Output: The sampled Model update $\Phi \Delta \mathbf{w}_t^k$

Compression ratio (r)	Algorithms	*Performance*		
		Accuracy	*Round*	*Cost (Megabit)*
0.05	FL-RND	0.73	192	8.52
	FL-FREQ	0.73	189	8.38
	FL-CS	0.82	200	8,87
0.1	FL-RND	0.78	200	17.74
	FL-FREQ	0.78	197	17.48
	FL-CS	0.85	199	17.65
0.2	FL-RND	0.82	200	35,49
	FL-FREQ	0.82	195	34,60
	FL-CS	0.87	193	34,24
1.0	FL-STD	0.87	191	169.44
1.0	Sign-SGD	0.85	200	332.67

Tab. 4.2: Summary of results on Fashion-MNIST dataset using the FL-CS scheme.

Compression ratio (r)	Algorithms	*Performance*			
		Bal_Acc	*AUROC*	*Round*	*Cost(Megabit)*
0.05	FL-RND	0.60	0.69	99	4.73
	FL-FREQ	0.69	0.76	100	4.78
	FL-CS	0.73	0.80	100	4.78
0.1	FL-RND	0.66	0.73	100	9.56
	FL-FREQ	0.71	0.78	100	9.56
	FL-CS	0.73	0.81	87	8.31
0.2	FL-RND	0.69	0.76	100	19.11
	FL-FREQ	0.72	0.80	100	19.11
	FL-CS	0.73	0.81	74	14.14
1.0	FL-STD	0.74	0.82	99	94.62
1.0	Sign-SGD	0.73	0.79	39	58.37

Tab. 4.3: Summary of results on Medical dataset using the FL-CS scheme.

FL-TOP: Federated Pruning for Compression

FL-TOP is inspired by the pruning techniques proposed in [Han+16] (see Section 3.2 in the Related Work for more details), and it aims to reduce the amount of parameters exchanged downstream (from the server to the participating entities) and upstream (from the participating entities to the server). In our scheme, each client updates only a small subset, Top-K, of the model parameters (weights) at each round. Only the K weight values of these Top-K parameters are updated during training, and neither the clients nor the server need to transfer the values of the remaining $n - K$ parameters, where n is the total number of parameters. The set of Top-K parameters do not change over the whole training

Algorithm 7: FL-TOP: Federated Learning

Server:
  Initialize common model w_0
  Select set $\mathbb{T}$ of Top-K updated weights' coordinates via public dataset
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\mathbf{c}_t^k = \mathbf{Client}_k(\mathcal{C}(\mathbf{w}_{t-1}, \mathbb{T}))$
    end
    $\mathbf{w}_t = \mathbf{w}_0$
    $j = 1$
    for *each coordinate* i *in* $\mathbb{T}$ **do**
      $\mathbf{w}_t[i] = \mathbf{w}_{t-1}[i] + \sum_k \frac{\mathbf{c}_t^k[j]}{|\mathbb{K}|}$
      $j = j + 1$
    end
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\hat{\mathbf{w}}_{t-1}^k)$**:**
  $\mathbf{w}_{t-1}^k = \mathbf{w}_0$
  $j = 1$
  for *each coordinate* i *in* $\mathbb{T}$ **do**
    $\mathbf{w}_{t-1}^k[i] = \hat{\mathbf{w}}_{t-1}^k[j]$
    $j = j + 1$
  end
  $\mathbf{w}_t^k = \mathbf{Top_k SGD}(D_k, \mathbf{w}_{t-1}^k, \mathbf{w}_0, T_{\text{gd}}, \mathbb{T})$
  Output: Model update $\mathcal{C}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k, \mathbb{T})$

Algorithm 8: $\mathbf{Top_k}$-Stochastic Gradient Descent

Input: D : training data, T_{gd} : local epochs, $\mathbf{w}$: weights, $\mathbf{w}_0$: first weights' initialization, $\mathbb{T}$: set of Top-K values coordinates .
for $t = 1$ **to** T_{gd} **do**
  Select batch $\mathbb{B}$ from D randomly
  $\mathbf{u} = -\eta \nabla f(\mathbb{B}; \mathbf{w})$
  for *each coordinate* i *in* $\mathbb{T}$ **do**
    $\mathbf{w}[i] = \mathbf{w}[i] + \mathbf{u}[i]$
  end
end
Output: Model $\mathbf{w}$

and are identical for all clients. We experimentally show in Section 5.4.5 that, if these K parameters are chosen carefully, the performance penalty is negligible even if $K = 0.005 \cdot n$, that is, 99.5% of the model parameters are pruned. Notice that unlike standard pruning techniques, where the set of pruned weights are re-selected after each SGD iteration [Han+16], our scheme always updates the same K parameters.

These Top-K parameters are selected by the server at the beginning of the protocol. More specifically, the server initializes the model and trains that with some public data that have a similar distribution as the clients' training data. After a few SGD iterations, the server selects the K parameters which values changed the most.

FL-TOP is described in Alg. 7. First, the server uses public data to identify the set $\mathbb{T}$ of the Top-K parameters ($K = |\mathbb{T}|$), before starting federated learning. In particular, starting from a public model $\mathbf{w}_0$, it accumulates the absolute value of gradients per parameter over T_{init} SGD iterations, and selects the K parameters with the largest accumulated gradients. After that, the values/updates[4] of these parameters are the only ones exchanged during the rest of the training between the server and the clients.

At each round, each selected client k uses the K updated weights $\hat{\mathbf{w}}_{t-1}$ received from the server to create a new weight vector $\mathbf{w}_{t-1}^k$ of size n, such that $\mathbf{w}_{t-1}^k$ is composed from the compressed vector $\hat{\mathbf{w}}_{t-1}^k$ of size $K \leq n$ (with coordinates in $\mathbb{T}$) and $n - K$ weights from the initialization vector

[4] weight values for downstream and update/gradients for upstream traffic

$\mathbf{w}_0$. $\mathbf{w}_0$ is identical for all participants and can be generated from a shared seed. Note that when $K = |\mathbb{T}| = n$, the scheme is equivalent to FL-STD. The weight vector $\mathbf{w}_{t-1}^k$ is used to train the client's model. However, only the weights in $\mathbb{T}$ are updated while the remaining ones are kept fixed. To do that, the weights not in $\mathbb{T}$ are reinitialized after each SGD iteration to $\mathbf{w}_0$. The server receives only the values from $\mathbf{w}_t^k - \mathbf{w}_{t-1}^k$ at coordinates $\mathbb{T}$, denoted by $\mathcal{C}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k)$ for short, from every client k, and updates the common model $\mathbf{w}_t$ with the average of these compressed updates (in Line 12).

4.2.2 Experimental Set-up

The goal of this section is to evaluate the performance of our proposed schemes FL-TOP on a benchmark dataset and a realistic in-hospital mortality prediction scenario. We aim at evaluating their performance with different levels of compression and comparing them with the performance of the following learning protocols[5]:

- FL-STD: The Standard Federated Learning scheme as described in Section 2.1 (see Alg. 1).
- FL-BASIC: A Federated Learning scheme that updates a random subset of parameters instead of the Top-K parameters at each SGD iteration. This subset is re-selected at the beginning of each new round. The $n - k$ non-selected parameters are still reinitialized after each SGD update as in FL-TOP.
- FL-CS: Our Federated Learning scheme that uses Compressive sensing (CS) to compress model updates from Section 4.1.4. See Section 2.3 for more details.
- FL-SIGN: Our sign-based Federated Learning scheme that uses quantization to compress model updates from Section 4.1.1.

Note that all compression operators in the baselines are linear, and hence they can also be used with secure aggregation.

We evaluate the above learning algorithms on the well-known Fashion-MNIST dataset and on the Premier Healthcare Database, described in Section 8.3.

Recall that the Top-K weights are selected before starting the federated learning process using public data. For Fashion-MNIST, we randomly select a batch with size 10 from MNIST dataset described in Section 8.3. For the medical dataset, we did not find any public dataset with the same features as ours, and for this reason, we selected randomly from the dataset a batch of 356 patients[6]. This set is used only by the server and never by any client. Afterwards, the server performs T_{init} SGD iterations starting from the model parameters $\mathbf{w}_0$ on the same batch to identify the Top-K weights. We experimentally show later that even these small batches are enough for the server to find a good set of Top-K weights.

More information about the hyper-parameter selection can be found in Tables 8.8,8.9, 8.10 in the Appendix. For FL-SIGN, γ is set to 0.001.

4.2.3 Performance Evaluation

Table 4.4 represents the best accuracy over 200 rounds for each scheme on the Fashion-MNIST dataset. $Round$ corresponds to the round when the best accuracy is reached and $Cost$ is the average bandwidth consumption calculated as: $r \times n \times 32 \times Round \times C$, where 32 is the number of bits necessary to represent a float value, n is the uncompressed model size, $r = \frac{|\mathbb{T}|}{n}$, $|\mathbb{T}|$ is the compressed model size, C is the sampling probability of a client, and $Round$ is the round when we get the the best accuracy.

Table 4.5 represents the best balanced accuracy over 100 rounds for each scheme on the Medical dataset. $AUROC$ (area under the receiver operating characteristic curve - see Section 8.4.2 for more details) corresponds to the $AUROC$ value when the best balanced accuracy is reached.

[5] More baselines are considered but due to the lack of space, We have decided to present only those which return the best results. All other results can be found in Tables 8.11,8.12,8.13.

[6] Reduced to 24 patients when we train via downsampling with 12 patients for each class

These tables show that the proposed non-private scheme FL-TOP has similar accuracy than the standard scheme FL-STD but reduces the bandwidth cost significantly. For example, with the Fashion-MNIST dataset, the FL-TOP accuracy reaches 0.85 when the compression ratio $r = 10\%$. In comparison, the standard FL-STD scheme reaches an accuracy of 0.86% but consumes 10 times more bandwidth. Furthermore, although FL-CS reaches the same accuracy than FL-TOP and consumes slightly less bandwidth upstream (9% less), its required downstream bandwidth is about 10 times larger (See Table 4.4 for more details). The results on the medical dataset are quite similar. In fact, FL-TOP achieves its best balanced accuracy (0.74) and AUROC (0.82) when $r = 10\%$ while the FL-STD scheme obtains similar performance but required about 11 times more upsteam and downstream bandwidth cost. FL-CS achieves similarly accuracy at $r = 10\%$ as FL-TOP but its downstream required bandwidth is about 11 times larger (see Table 4.5 for more details).

Although, FL-SIGN reaches the same accuracy (0.84) on Fashion-MNIST than FL-TOP (at r=0.5%) and with less upstream costs (38% less), however, its required downstream bandwidth is about 20 times more than FL-TOP. Similarly, on the medical dataset, FL-SIGN reaches the same accuracy compared to FL-TOP at $r = 0.5\%$ and with less downstream bandwidth but much more upstream bandwidth (see Table 4.5).

r	Algorithms	*Performance*			
		Accuracy	*Round*	*Downstream Cost* (Kilobyte)	*Upstream Cost* (Kilobyte)
0.5%	FL-BASIC	0.65	193	21402.03	107
	FL-CS	0.57	185	20514.9	102.56
	FL-TOP	**0.82**	200	**110.88**	**110.88**
5%	FL-BASIC	0.78	196	21734.70	1086.73
	FL-CS	0.82	200	22178.27	1108.91
	FL-TOP	**0.84**	200	**1108.91**	**1108.91**
10%	FL-BASIC	0.81	196	21734.70	2173.47
	FL-CS	0.85	182	20182.22	2018.22
	FL-TOP	**0.85**	199	**2206.74**	**2206.74**
100%	FL-STD	0.86	200	22178.27	22178.27
	FL-SIGN	0.84	197	21845.59	682.67

Tab. 4.4: Summary of results on Fashion-MNIST dataset using the FL-TOP scheme.

r	Algorithms	*Performance*				
		Bal_Acc	*AUROC*	*Round*	*Downstream Cost* (Kilobyte)	*Upstream Cost* (Kilobyte)
0.1%	FL-BASIC	0.51	0.51	99	11829.42	11.82
	FL-CS	0.53	0.55	100	11948.91	11.94
	FL-TOP	**0.69**	**0.76**	68	**8.12**	**8.12**
5%	FL-BASIC	0.72	0.80	100	11948.91	597.45
	FL-CS	0.73	0.81	98	11709.93	585.5
	FL-TOP	**0.72**	**0.80**	95	**567.57**	**567.57**
10%	FL-BASIC	0.74	0.81	100	11948.91	1194.89
	FL-CS	0.74	0.82	100	11948.91	1194.89
	FL-TOP	**0.74**	**0.82**	90	**1075.40**	**1075.40**
100%	FL-STD	0.74	0.82	99	11829.42	11829.42
	FL-SIGN	0.72	0.79	68	8125.26	253.91

Tab. 4.5: Summary of results on Medical dataset using the FL-TOP scheme.

4.3 Conclusion

In this Chapter, we have shown that it is possible to have bandwidth efficient Federated Learning solutions which are as accurate as the standard Federated Learning scheme (FL-STD). Indeed, we have proposed two approaches which compress either the updates or the model.

The approach of reducing bandwidth by compressing the updates includes two schemes: 1) FL-SIGN is based on quantization and is almost as accurate as FL-STD but incurs less communication overhead by reducing the number of bits per float value from 32 bits to 1 bit during upstream step (from the clients to the server); 2) FL-CS allows to reduce the upstream bandwidth cost by up to 95% by exploiting the sparsity of the model updates via Compressive Sensing theory.

The approach of reducing bandwidth by reducing the model includes one scheme called FL-TOP, which reduces both upstream and downstream bandwidth costs up to 99.9%. Therefore, it is particularly adapted to bandwidth and energy constrained applications as it is the case for mobile systems.

5 Designing Private and Bandwidth Efficient Federated Learning Schemes

Although Federated Learning improves privacy, model parameters can leak information about the training data. Indeed, [ZLH19; ZMB20; GBDM20] presented some attacks that allow an adversary to reconstruct pieces of the training data of some entities. [NSH19; MSDS19] define a membership attack that allows to infer if a particular record is included in the data of a specific entity. Similarly, [MSDS19] define an attack which aims at inferring if a subgroup of people with a specific property, like for instance the skin color or ethnicity, is included in the dataset of a particular participating entity. A solution to prevent these attacks and provide theoretical guarantees is to use a privacy model called Differential Privacy [DR14].

We propose differentially private extensions of the three schemes proposed in Chapter 4, namely FL-SIGN-DP, FL-CS-DP and FL-TOP-DP. For each proposed scheme, we follow an approach where clients themselves add noise in a distributed manner so that the aggregated updates are sufficiently noised to have meaningful differential privacy. To this end, individual noisy updates are encrypted with a simple and efficient encryption scheme taken from [ÁC11].

- FL-SIGN-DP: In order to diminish the communication costs of our DP algorithm, we propose a novel discretized and distributed version of the Gaussian Mechanism. In particular, as opposed to the standard Gaussian Mechanism [14], the noise values come from a discretized domain and are tightly concentrated around its mean depending on the desired privacy guarantee ϵ. As a result, these values can be encoded with fewer bits than if they came from a continuous Gaussian distribution.

- FL-CS-DP reduces the added Gaussian noise by reducing the sensitivity of the model via compression. Indeed, it adds the noise only to the first coefficients of the discrete cosine transform of each update. Because of the large energy compaction property of DCT, the first coefficients, which correspond to the low frequency components of the update, tend to have the largest magnitude and hence convey most information about the model update. This leads to larger value-to-noise level and therefore better performance. Also, reconstructing the approximated gradients is an instance of Basis Pursuit Denoising (or LASSO), which can be solved with efficient solvers that provide large accuracy despite the added Gaussian noise. Finally, we show that FL-CS-DP produces more accurate and bandwidth efficient models than FL-STD-DP, that is, the differentially private variant of the vanilla Federated Learning protocol without any compression.

- FL-TOP-DP updates for its part only few weights during the whole training process while keeping the remaining ones constant (defined as a constraint). As a result, it reduces the sensitivity of the model. Moreover, we observed experimentally that the constraint leads to greater importance of the updated weights by increasing their values. This result is important when differential privacy is used as it leads to larger value-to-noise level and hence better performance.

Notice that the secure aggregation protocol can be used only with linear compression schemes such as the ones we propose.

5.1 Privacy Model

We consider an adversary, or a set of colluding adversaries, who can access any update vector sent by the server or any clients at each round of the protocol. A plausible adversary is a participating entity, i.e. a malicious client or server, that wants to infer the training data used by other participants. The adversary is *passive* (i.e., honest-but-curious), that is, it follows the learning protocol faithfully.

Different privacy requirements can be considered depending on what information the adversary aims to infer. In general, private information can be inferred about:

- any record (user) in any dataset of any client (*record-level privacy*),
- any client/party (*client-level privacy*).

To illustrate the above requirements, suppose that several banks build a common model to predict the creditworthiness of their customers. A bank certainly does not want other banks to learn the financial status of any of their customers (record privacy) and perhaps not even the average income of all their customers (client privacy).

Record-level privacy is a standard requirement used in the privacy literature and is usually weaker than client-level privacy. Indeed, client-level privacy requires to hide any information which is unique to a client including perhaps all its training data.

5.2 FL-SIGN-DP: Private and Bandwidth-Efficient Federated Learning via Quantization

In FL-SIGN, a participant only sends the signs of its updates, as opposed to their actual value, hence it intuitively reveals less information about the client's dataset than the original FL-STD scheme. In order to experimentally validate this intuition, we implemented the inference attack described in [MSDS19] on FL-STD and FL-SIGN[1]. The results clearly validated our intuition (the attack accuracy dropped from 92% for FL-STD to 50% for FL-SIGN). Although these results are very promising and might confirm that privacy is preserved in practice, it does not provide any provable privacy-preserving guarantees. In order to obtain theoretically, backed and proven private schemes, we extend FL-SIGN with Differential Privacy. Our goal is to design differentially private schemes that are efficient in terms of accuracy and bandwidth (even for small ε values).

5.2.1 Operation

We aim at developing a solution that provides *client-level privacy and is also bandwidth efficient*. For example, in the scenario of collaborating banks, we aim at protecting any information that is unique to each single bank's training data. The adversary should not be able to learn from the received model or its updates whether any client's data is involved in the federated run (up to ε and δ). We believe that this adversarial model is reasonable in many practical applications when the confidential information spans over multiple samples in the training data of a single client (e.g., the presence of a group a samples, such as people from a certain race). Differential Privacy guarantees plausible deniability not only to any groups of samples of a client but also to any client in the federated run. Therefore, any negative privacy impact on a party (or its training samples) cannot be attributed to their involvement in the protocol run.

To guarantee differential privacy per client, every client should add enough noise to its update locally such that the server cannot learn any client-specific information from the noisy update. However, this approach (aka, local differential privacy [EPK14]) requires so much perturbation that it is impractical if the number of clients is limited. Instead, likewise [Tru+19], we follow a different approach where clients themselves add noise in a distributed manner so that the aggregated updates are sufficiently noised to have meaningful differential privacy. To this end, individual noisy updates are encrypted with a simple and efficient encryption scheme taken from [ÁC11; Bon+16]. The purpose of this encryption is to prevent the adversary from accessing the individual (and weakly-noised) update

[1] A model is trained for gender classification on the LFW dataset. The adversary's goal is to infer from the model updates whether a specific group of individuals in a client's dataset are black.

per client but only their sum over all clients which is in turn sufficiently noised to guarantee DP for any client.

Specifically, each client k first computes the gradient update $\Delta \mathbf{w}_t^k$ (in Line 12 of Alg. 9) and then takes the sign vector of this update. Then, a random noise share ρ_k is added to the sign vector $\mathsf{sign}(\Delta \mathbf{w}_t^k)$ so that $\sum_{k\in\mathbb{K}} \mathsf{sign}(\Delta \mathbf{w}_t^k) + \rho_k$ satisfies differential privacy. A simple solution is that $\rho_i \sim \mathcal{G}(0, \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})$, which means that $\sum_{k\in\mathbb{K}} \mathsf{sign}(\Delta \mathbf{w}_t^k) + \sum_{k\in\mathbb{K}} \rho_k = \sum_{k\in\mathbb{K}} \mathsf{sign}(\Delta \mathbf{w}_t^k) + \mathcal{G}(0, \sqrt{n}\mathbf{I}\sigma)$ as the sum of Gaussian random variables also follows Gaussian distribution[2]. Indeed, the variance of the Gaussian noise has to be proportional to the L_2-sensitivity of the sign vector which is no more than $\sqrt{n}$, where n is the number of parameters.

However, recall that the adversary can access $\mathsf{sign}(\Delta \mathbf{w}_t^k) + \rho_k$, which means that, if $|\mathbb{K}|$ is too large, ρ_k is likely to be small allowing the adversary to learn $\mathsf{sign}(\Delta \mathbf{w}_t^k) + \rho_k$ very accurately. For this reason, each client k encrypts $\mathsf{sign}(\Delta \mathbf{w}_t^k) + \rho_k$ and sends the encrypted result to the aggregator. After summing all the encrypted values, the server obtains $\sum_k \mathsf{Enc}_{\mathbf{K}_k}(\mathsf{sign}(\Delta \mathbf{w}_t^k) + \rho_k) = \sum_k \mathsf{sign}(\Delta \mathbf{w}_t^k) + \mathcal{G}\left(0, \sqrt{n}\sigma\mathbf{I}\right)$ where $\mathsf{Enc}_{\mathbf{K}_k}(\mathsf{sign}(\Delta \mathbf{w}_t^k) + \rho_k) = \mathsf{sign}(\Delta \mathbf{w}_t^k) + \rho_k + \mathbf{K}_k \mod m$ and $\sum_k \mathbf{K}_k = \mathbf{0}$ (see [ÁC11; Bon+16] for details). Here, modulo is taken element-wise and $m = 2^{\lceil \log_2(\max_k ||1+\rho_k||_\infty |\mathbb{K}|)\rceil}$. Therefore, the server can only access the aggregate which is sufficiently noised to guarantee differential privacy; any client-specific information that could be learnt from the noisy aggregate is quantified by the moments accountant described in Section 2.2. To make learning more resilient to perturbation, the server takes the sign of the sum of updates and scales the result with $\gamma < 1$ which is crucial to achieve convergence in practice especially if $\sqrt{n}\sigma$ is large.

Unfortunately, the above simple approach is not bandwidth efficient; adding noise from the continuous domain requires each noisy update $\mathsf{sign}(\Delta \mathbf{w}_t^k) + \rho_k$ to be encoded as a floating-point number[3] (represented by at least 32 bits on a commodity hardware) no matter that $\mathsf{sign}(\Delta \mathbf{w}_t^k)$ would need only 1 bit per coordinate. Therefore, the noisy update needs at least 32 times more data to be transferred from a client to the server than with FL-SIGN (in Alg. 3).

To alleviate the above bandwidth problem, each client k generates a random integer from a *discrete* Gaussian distribution with mean $\mathsf{sign}(\Delta \mathbf{w}_t^k)$, encrypts this random integer, and sends the result for aggregation. Since the discrete Gaussian random variable has an integer value and is concentrated around its mean, its value can be encoded with fewer bits than a floating-point number. The new learning algorithm, called FL-SIGN-DP, guarantees differential privacy for any client and is summarized in Alg. 9.

In what follows, we first describe the Discrete Gaussian Mechanism (DGM), which is used in FL-SIGN-DP, and prove that it practically provides the same privacy guarantee as the continuous Gaussian Mechanism (GM) if its variance is sufficiently large. This allows us to precisely quantify the privacy guarantee of FL-SIGN-DP. Finally, we show that using DGM instead of (continuous) GM in FL-SIGN-DP reduces the communication overhead by roughly 40%.

5.2.2 Discrete Gaussian Mechanism (DGM)

The discrete Gaussian distribution has probability mass function

$$\mathsf{pmf}_{\mathcal{DG}(\mu,\xi)}(x) = Z^{-1} \exp(-(x-\mu)^2/2\xi^2) \tag{5.1}$$

where $Z = \sum_{x\in\mathbb{Z}} \exp(-(x-\mu)^2/2\xi^2)$. Note that $\mu \in \mathbb{R}$ but the support of $\mathcal{DG}$ is always $\mathbb{Z}$. Although Z is infeasible to compute, there are several efficient techniques to sample from the discrete Gaussian distribution [MW17].

The next lemma shows that the pmf of the discrete Gaussian distribution can be almost perfectly approximated by its continuous counterpart if ξ is large enough.

Lemma 1. *Let* $\mathsf{pmf}_{\mathcal{DG}(\mu,\xi)}(x)$ *and* $\mathsf{pdf}_{\mathcal{G}(\mu,\xi)}(x)$ *be as defined in Eq.* (5.1) *and Eq.* (2.1)*, respectively, and* $\kappa(\xi) = \frac{2e^{-2\pi^2\xi^2}}{1-e^{-6\pi^2\xi^2}}$*. Then,* $1-\kappa(\xi) \le \frac{\mathsf{pdf}_{\mathcal{G}(\mu,\xi)}(x)}{\mathsf{pmf}_{\mathcal{DG}(\mu,\xi)}(x)} \le 1+\kappa(\xi)$ *for* $x \in \mathbb{Z}$.

The proof can be found in Appendix 8.1.1.

[2] More precisely, $\sum_i \mathcal{G}(\nu_i, \xi_i) = \mathcal{G}\left(\sum_i \nu_i, \sqrt{\sum_i \xi_i^2}\right)$

[3] and then as a large integer for encryption

The multivariate spherical version of $\mathcal{DG}$ can be defined analogously to the spherical Gaussian distribution, that is, if $\mathbf{z} \sim \mathcal{DG}(\boldsymbol{\mu}, \xi)$, then $z_i \sim \mathcal{DG}(\mu_i, \xi)$ independently for each i.

The Discrete Gaussian Mechanism (DGM) is defined analogously to the (continuous) Gaussian Mechanism except that it uses discrete Gaussian noise instead of its continuous counterpart for perturbation. The next theorem shows that the moments of DGM can be tightly upper bounded by that of the continuous Gaussian mechanism if ξ is large enough, and hence the privacy guarantee of DGM can be efficiently and accurately approximated.

Let $\eta_0^{\mathcal{G}}(x|\xi) = \mathsf{pdf}_{\mathcal{G}(0,\xi)}(x)$ and $\eta_1^{\mathcal{G}}(x|\xi) = (1-C)\mathsf{pdf}_{\mathcal{G}(0,\xi)}(x) + C\mathsf{pdf}_{\mathcal{G}(1,\xi)}(x)$ where C is the sampling probability of a single client in a single round. Let

$$\alpha_{\mathcal{G}}(\lambda|C) = \log \max(E_1(\lambda, \xi, C), E_2(\lambda, \xi, C)) \tag{5.2}$$

where $E_1(\lambda, \xi, C) = \int_{\mathbb{R}} \eta_0^{\mathcal{G}}(x|\xi, C) \cdot \left(\frac{\eta_0^{\mathcal{G}}(x|\xi,C)}{\eta_1^{\mathcal{G}}(x|\xi,C)}\right)^{\lambda} dx$ and $E_2(\lambda, \xi, C) = \int_{\mathbb{R}} \eta_1^{\mathcal{G}}(x|\xi, C) \cdot \left(\frac{\eta_1^{\mathcal{G}}(x|\xi,C)}{\eta_0^{\mathcal{G}}(x|\xi,C)}\right)^{\lambda} dx$. $\alpha_{\mathcal{DG}}(\lambda|C)$ is defined analogously to $\alpha_{\mathcal{G}}(\lambda|C)$.

Theorem 2 (Privacy of DGM). *$\alpha_{\mathcal{DG}}(\lambda|C) \leq \alpha_{\mathcal{G}}(\lambda|C) + \log\left(\frac{(1+\kappa(\xi))^{\lambda}}{(1-\kappa(\xi))^{\lambda+1}}\right)$ for any C, where $\kappa(\xi)$ is defined in Lemma 1. Therefore, DGM is $(\min_{\lambda}(\left(\alpha_{\mathcal{G}}(\lambda|C) + \log\left(\frac{(1+\kappa(\xi))^{\lambda}}{(1-\kappa(\xi))^{\lambda+1}}\right)\right) - \log\delta)/\lambda, \delta)$-DP.*

The proof can be found in Appendix 8.1.2. Given a fixed value of δ, ε is computed numerically as in [Aba+16; MTZ19].

Notice that, in [CKS20], the privacy bound of the proposed Discrete Gaussian Mechanism is same as the one of its continuous counterpart. However, to obtain this result, they considered a Discrete Gaussian Mechanism with an integer mean. Therefore, their mechanism can only be used with integer values, which is not always the case in practice. Hence, our result is more general and can even be used with float values.

5.2.3 Privacy of FL-SIGN-DP

As shown in Alg. 9, each client k generates a random integer vector $\mathbf{z}_k \sim \mathcal{DG}(\mathsf{sign}(\Delta\mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})$ in FL-SIGN-DP. Then, every client sends the encrypted result $\mathsf{Enc}_{\mathbf{K}_k}(\mathbf{z}_k)$ to the aggregator. After summing all the encrypted integers, the server obtains

$$\sum_k \mathsf{Enc}_{\mathbf{K}_k}(\mathbf{z}_k) = \sum_k \mathbf{z}_k = \sum_k \mathcal{DG}(\mathsf{sign}(\Delta\mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|}) \tag{5.3}$$

The next theorem, proved in Appendix 8.1.3, shows that FL-SIGN-DP is differentially private, supposing that the adversary can only access $\sum_k \mathcal{DG}(\mathsf{sign}(\Delta\mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})$ except any of its members $\mathcal{DG}(\mathsf{sign}(\Delta\mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})$.

Theorem 3 (Privacy of FL-SIGN-DP). *For any $\delta > 0$, FL-SIGN-DP is $(\min_{\lambda}(T\cdot\left(\alpha_{\mathcal{G}}(\lambda|C) + \log\left(\frac{(1+\kappa(\sqrt{n}\sigma))^{\lambda}}{(1-\kappa(\sqrt{n}\sigma))^{\lambda+1}}\left(\frac{1+\nu}{1-\nu}\right)^3\right)\right) - \log\delta)/\lambda, \delta)$-DP, where $\sigma \geq \sqrt{|\mathbb{K}|\ln(2+2/\nu)/2n\pi^2}$ and κ is defined in Lemma 1.*

Again, given a fixed value of δ, ε is computed numerically as in [Aba+16; MTZ19].

5.2.4 Communication Overhead

The domain of $\mathbf{z}$ in Eq. (5.3) is the support of $\mathcal{DG}$ which is still unbounded. This means that the size of the encrypted text can be very large though with exponentially small probability. Indeed, $||\mathbf{z}_k||_\infty$ is unbounded and hence modulo $m = 2^{\lceil \log_2(\max_k ||\mathbf{z}_k||_\infty |\mathbb{K}|)\rceil}$ has to be large. To overcome this problem, we choose modulo m to be so large that the probability that $2^{\lceil \log_2(\max_k ||\mathbf{z}_k||_\infty |\mathbb{K}|)\rceil}$ is larger than m is negligible. For this purpose, we rely on the following concentration inequality of the discrete Gaussian distribution.

Lemma 2 ([MW17], Lemma 2.2). *For any $\nu > 0$, $\xi > \sqrt{\ln(2+2/\nu)/2\pi^2}$, and $t > 0$, $\Pr_{x\sim\mathcal{DG}(\mu,\xi)}[|x-\mu| \geq t\cdot\xi] \leq 2e^{-t^2/2}\cdot\frac{1+\nu}{1-\nu}$.*

Lemma 2 implies that if $\xi = \sqrt{n}\sigma/\sqrt{|\mathbb{K}|} > 3.51$ then $\frac{1+\nu}{1-\nu} < \frac{3}{2}$ and $\Pr_{\mathbf{z}\sim\mathcal{DG}(\boldsymbol{\mu}, \sqrt{n}\sigma\mathbf{I}/\sqrt{|\mathbb{K}|})}[||\mathbf{z} - \boldsymbol{\mu}||_\infty \geq t\sqrt{n}\sigma] \leq 3ne^{-|\mathbb{K}|t^2/2}$ after applying the union bound. For example, if $m = 2^{\lceil \log_2(12\sqrt{n}\sigma|\mathbb{K}|)\rceil}$

Algorithm 9: FL-SIGN-DP: Federated Learning with Client Privacy

Server:
  Initialize common model w_0
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients randomly
    for *each client* k *in* $\mathbb{K}$ **do**
      $\Delta \mathbf{w}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \gamma \cdot \text{sign}\left(\sum_k \Delta \mathbf{w}_t^k\right)$
  end
$\mathbf{Client}_k(\mathbf{w})$**:**
  $\mathbf{w}_{t-1}^k = \mathbf{w}$
  $\Delta \mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}}) - \mathbf{w}_{t-1}^k$
  Output: $\text{Enc}_{K_k}\left(\mathcal{DG}\left(\text{sign}\left(\Delta \mathbf{w}_t^k\right), \sqrt{n}\mathbf{I}\sigma/\sqrt{|K|}\right)\right)$

(i.e., $t = 12$) then the probability that $||\mathbf{z}_k - \boldsymbol{\mu}_k||_\infty$ cannot be bounded by $12\sqrt{n}\sigma$ per client is less than 2^{-80} even if $|\mathbb{K}| = 1$ and $n = 10^7$. Thus, a client needs to transfer $n \cdot \log_2\left(2^{\lceil \log_2((12\sqrt{n}\sigma + \max_k ||\boldsymbol{\mu}_k||_\infty)|\mathbb{K}|)\rceil}\right)$ bits in total to the aggregator. For example, if $|\mathbb{K}| = 100$, $\max_k ||\boldsymbol{\mu}_k||_\infty = 1$, $\sigma = 1$ (i.e., $\varepsilon \approx 0.2$), then $\log_2 m = 22$. By contrast, if noise was generated from the continuous domain, then $\log_2 m = 32$ which means that DGM reduces the communication overhead by roughly 32%.

Notice that if ε or δ is smaller (i.e., there is stronger privacy guarantee), then σ is larger which implies that m also increases, and hence more bits need to be transferred to the server per parameter. This results in a trade-off between Differential Privacy and bandwidth efficiency.

Algorithm 10: FL-STD-DP: Federated Learning with Client Privacy

Server:
  Initialize common model w_0
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients randomly
    for *each client* k *in* $\mathbb{K}$ **do**
      $\Delta \tilde{\mathbf{w}}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \frac{1}{|\mathbb{K}|}\sum_k \Delta \tilde{\mathbf{w}}_t^k$
  end
$\mathbf{Client}_k(\mathbf{w})$**:**
  $\mathbf{w}_{t-1}^k = \mathbf{w}$
  $\Delta \mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}}) - \mathbf{w}_{t-1}^k$
  $\Delta \hat{\mathbf{w}}_t^k = \Delta \mathbf{w}_t^k / \max\left(1, \frac{||\Delta \mathbf{w}_t^k||_2}{S}\right)$
  Output: $\text{Enc}_{K_k}(\mathcal{G}(\Delta \hat{\mathbf{w}}_t^k, S\mathbf{I}\sigma/\sqrt{|K|}))$

5.2.5 Robustness of FL-SIGN-DP Against non-adversarial Client Failures

If any client fails to add its noise share to the model update for any reason, the aggregate will not have sufficient amount of noise to guarantee differential privacy. A straightforward countermeasure is to increase the variance of the added noise so that even if r clients fail, the sum of $CN - r$ noise shares are still enough for differential privacy. In particular, each client k sends $\text{Enc}_{K_k}(\mathcal{DG}(\text{sign}(\Delta \mathbf{w}_t^k), \sqrt{n}\sigma\mathbf{I}/\sqrt{CN - r}))$ to the server for aggregation. Obviously, if less than r nodes fail, the aggregate will have larger noise than what is necessary for differential privacy.

5.2.6 Experimental Set-up

The performance of FL-SIGN-DP is compared with FL-STD-DP in Table 5.1 and 5.2. FL-STD-DP is an extension of FL-STD to provide client-level differential privacy. Specifically, in FL-STD-DP, the randomly selected clients first clip their model update vector to have a bounded L_2-norm[4], add continuous Gaussian noise to the clipped update vector, and then transfer the *non-quantized* noisy model update

[4]The sensitivity $S = \sqrt{n}$ and $\gamma = 0.005$ for FL-SIGN-DP. For FL-STD-DP, the server computes the median L_2-norm value over N L_2-norm values received during an additional initialization round. Hence, S is set to 1.73 and 2.15 for MNIST and Fashion-MNIST, respectively.

to the server (see Alg. 10 for more details). The configurations of these protocols are summarized in Table 8.2.

5.2.7 Performance Evaluation

Table 5.1 and 5.2 show the best model accuracy observed over 200 rounds with each algorithm on the MNIST and Fashion-MNIST datasets, respectively. FL-STD-DP provides the best accuracy; for MNIST, it is 86-93%, and for Fashion-MNIST, it is 61-78% depending on the privacy parameter ε. The performance degradation of FL-SIGN-DP compared to FL-STD-DP is at most 0.02 on MNIST and 0.07 on Fashion-MNIST. In fact, FL-SIGN-DP outperforms FL-STD-DP for small privacy budget (i.e., $\varepsilon = 1$). As expected, weaker privacy requirement (i.e., larger ε) needs smaller noise magnitude and hence better accuracy for all algorithms.

The communication cost of FL-SIGN-DP is 66% of that of FL-STD-DP. Specifically, while FL-STD-DP needs 32 bits per parameter, FL-SIGN-DP requires 21-22 bits depending on the value of ε[5]. If ε is smaller, the variance of the noise is larger, and hence more bits are necessary to encode the noisy signs.

The convergence rates of FL-SIGN and FL-SIGN-DP are $O\left(\frac{1}{\sqrt{T_{\mathsf{cl}}CN}}\right)$ and $O\left(\frac{1}{\sqrt{T_{\mathsf{cl}}CN}} + \frac{\sqrt{3n}\sigma}{\sqrt{T_{\mathsf{cl}}CN}}\right)$, respectively, supposing that $\gamma = O(1/\sqrt{T_{\mathsf{cl}}})$, $T_{\mathsf{gd}} = 1$, $|\mathbb{B}| = T_{\mathsf{cl}}$ (see Appendix 8.1.4 for the proofs). Therefore, the "cost of privacy" in convergence rate is $O\left(\frac{\sqrt{3n}\sigma}{\sqrt{T_{\mathsf{cl}}CN}}\right)$ which is due to the added noise.

	$\varepsilon = 1$		$\varepsilon = 2$		$\varepsilon = 4$	
	Acc	Cost	Acc	Cost	Acc	Cost
FL-STD-DP	0.86	32	0.92	32	0.93	32
FL-SIGN-DP	0.87	22	0.90	21	0.91	21

Tab. 5.1: Model accuracy and communication cost on MNIST dataset using FL-SIGN-DP. We give the communication cost per parameter value (bits/parameter) for any value of ε.

	$\varepsilon = 1$		$\varepsilon = 2$		$\varepsilon = 4$	
	Acc	Cost	Acc	Cost	Acc	Cost
FL-STD-DP	0.61	32	0.74	32	0.78	32
FL-SIGN-DP	0.63	22	0.70	21	0.73	21

Tab. 5.2: Model accuracy and communication cost with Fashion-MNIST dataset using FL-SIGN-DP. We give the communication cost per parameter value (bits/parameter) for any value of ε.

5.3 FL-CS-DP: Private and Bandwidth-Efficient Federated Learning via Compressive Sensing

FL-CS-DP is the private extension of FL-CS described in Section 4.1.4. It provides the same guarantee as FL-SIGN-DP and considers the same *Privacy model* (see Section 5.1 and Section 5.2.1 for more details). Therefore, it aims at preserving the privacy of each client instead of each record (client-level privacy). In what follows, we first describe the operation required to reach this guarantee and then we evaluate our private scheme on both: accuracy and bandwidth efficiency aspects.

5.3.1 Operation

FL-CS-DP is described in Alg. 11. Client-level differential privacy requires each client to add Gaussian noise to the compressed model updates. In particular, each client first calculates $\mathbf{c}_t^k = \mathcal{C}(\Delta\mathbf{w}_t^k, m)$ (in Line 19), which is then clipped (in Line 20) to obtain $\hat{\mathbf{c}}_t^k$ with L_2-norm at most S. Then, random noise $\mathbf{z}_k \sim \mathcal{G}(0, S\sigma\mathbf{I}/\sqrt{\mathbb{K}})$ is added to $\hat{\mathbf{c}}_t^k$ such that $\sum_{k\in\mathbb{K}}(\hat{\mathbf{c}}_t^k + \mathbf{z}_k) = \sum_{k\in\mathbb{K}} \hat{\mathbf{c}}_t^k + \mathcal{G}(0, S\sigma\mathbf{I})$ as the sum of Gaussian random variables also follows Gaussian distribution[6] and then differential privacy is satisfied where ε and δ can be computed using the moments accountant described in Section 2.2.

[5] It is computed from $\log_2\left(2^{\lceil \log_2((12\sqrt{n}\sigma + \max_k ||\boldsymbol{\mu}_k||_\infty)|\mathbb{K}|)\rceil}\right)$ where σ is obtained from ε and $\delta = 10^{-5}$ using the moments accountant. This ensures that the magnitude of the noisy update per model parameter is less than the modulus n with probability at most 2^{-80} (see Section 5.2.4).

[6] More precisely, $\sum_i \mathcal{G}(\nu_i, \xi_i) = \mathcal{G}(\sum_i \nu_i, \sqrt{\sum_i \xi_i^2})$

However, as the noise is inversely proportional to $\sqrt{\mathbb{K}}$, $\mathbf{z}_k$ is likely to be small if $|\mathbb{K}|$ is too large. Therefore, the adversary accessing an individual update $\hat{\mathbf{c}}_t^k + \mathbf{z}_k$ can almost learn a non-noisy update since $\mathbf{z}_k$ is small. Hence, each client uses secure aggregation to encrypt its individual update before sending it to the server. Upon reception, the server sums the encrypted updates as:

$$\begin{aligned}\sum_{k\in\mathbb{K}} \mathbf{y}_t^k &= \sum_{k\in\mathbb{K}} \mathsf{Enc}_{K_k}(\hat{\mathbf{c}}_t^k + \mathbf{z}_k) \\ &= \sum_{k\in\mathbb{K}} \hat{\mathbf{c}}_t^k + \sum_{k\in\mathbb{K}} \mathbf{z}_k \\ &= \sum_{k\in\mathbb{K}} \hat{\mathbf{c}}_t^k + \mathcal{G}(0, S\sigma\mathbf{I}) \end{aligned} \tag{5.4}$$

where $\mathsf{Enc}_{K_k}(\hat{\mathbf{c}}_t^k + \mathbf{z}_k) = \hat{\mathbf{c}}_t^k + \mathbf{z}_k + \mathbf{K}_k \mod p$ and $\sum_k \mathbf{K}_k = 0$ (see [ÁC11; Bon+16] for details). Here the modulo is taken element-wise and $p = 2^{\lceil \log_2(\max_k ||\hat{\mathbf{c}}_t^k + \mathbf{z}_k||_\infty |\mathbb{K}|)\rceil}$. Let $\gamma_t^k = 1/\max\left(1, \frac{||\mathbf{c}_t^k||_2}{S}\right)$. Then,

$$\begin{aligned}\sum_{k\in\mathbb{K}} \hat{\mathbf{c}}_t^k &= \sum_{k\in\mathbb{K}} \gamma_t^k \mathbf{c}_t^k \\ &= \sum_{k\in\mathbb{K}} \gamma_t^k \mathcal{C}(\Delta\mathbf{w}_t^k, m) \\ &= \mathcal{C}(\sum_{k\in\mathbb{K}} \gamma_t^k \Delta\mathbf{w}_t^k, m) \end{aligned} \tag{5.5}$$

where the last equality comes from the linearity of the compression operation (see Section 2.3). Plugging Eq. (5.5) into Eq. (5.4). we get that

$$\sum_{k\in\mathbb{K}} \mathbf{y}_t^k = \mathcal{C}(\sum_{k\in\mathbb{K}} \gamma_t^k \Delta\mathbf{w}_t^k, m) + \mathcal{G}(0, S\sigma\mathbf{I})$$

This is an instance of BPDN (see Section 2.3), and therefore the direct reconstruction of $\sum_{k\in\mathbb{K}} \mathbf{y}_t^k$ would be an approximation of $\sum_{k\in\mathbb{K}} \gamma_t^k \Delta\mathbf{w}_t^k$. However, analogously to FL-CS, the server applies error propagation and computes the (noisy) error $\mathbf{e}_t$ from $\mathbf{y}_t = (1/|\mathbb{K}|)\sum_{k\in\mathbb{K}} \mathbf{y}_t^k$ (in Line 10), and decompresses $\mathbf{e}_t$ into $\mathbf{s}_t$ by using OWL-QN. Recall that the reconstruction algorithm solves the BPDN problem, where a sparse vector $\mathbf{s}$ is reconstructed from m noisy measurements of the form $\Theta\mathbf{s} + \mathbf{z}$, where the noise $\mathbf{z} \in \mathbb{R}^m$ is assumed to be identically and independently distributed over its elements with a Gaussian distribution [JV10; CDS01]. Since $z \sim \mathcal{G}(0, S\mathbf{I}\sigma)$ in our case, the reconstruction algorithm is therefore optimized to reconstruct the differentially private compressed vectors (see Theorem 1).

5.3.2 Privacy of FL-CS-DP

Privacy Analysis The server can only access the noisy aggregate which is sufficiently perturbed to ensure differential privacy; any client-specific information that could be inferred from the noisy aggregate is tracked and quantified by the moments accountant, described in Section 2.2, as follows.

Let $\eta_0(x|\xi) = \mathsf{pdf}_{\mathcal{G}(0,\xi)}(x)$ and $\eta_1(x|\xi) = (1-C)\mathsf{pdf}_{\mathcal{G}(0,\xi)}(x) + C\mathsf{pdf}_{\mathcal{G}(1,\xi)}(x)$ where C is the sampling probability of a single client in a single round. Let

$$\alpha(\lambda|C) = \log\max(E_1(\lambda,\xi,C), E_2(\lambda,\xi,C)) \tag{5.6}$$

where $E_1(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_0(x|\xi,C) \cdot \left(\frac{\eta_0(x|\xi,C)}{\eta_1(x|\xi,C)}\right)^\lambda dx$ and $E_2(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_1(x|\xi,C) \cdot \left(\frac{\eta_1(x|\xi,C)}{\eta_0(x|\xi,C)}\right)^\lambda dx$.

Theorem 4 (Privacy of FL-CS-DP)**.** *FL-CS-DP is* $(\min_\lambda(T_{\mathsf{cl}} \cdot \alpha(\lambda|C) - \log\delta)/\lambda, \delta)$*-DP.*

Given a fixed value of δ, ε is computed numerically as in [Aba+16; MTZ19].

The magnitude of the added Gaussian noise is proportional to the sensitivity S, which is in turn often proportional to the model size n [Zhu+20]. Hence, when n becomes large, SGD often fails to

converge due to the perturbation error caused by the added noise [Zhu+20]. In our approach, the perturbation error is less since Gaussian noise is added to the compressed vector with size $m < n$. On the other hand, compression also induces some reconstruction error owing to its lossy nature. The total error is the sum of the reconstruction and the perturbation error and is quantified in Theorem 1. Finding the right trade-off between these two errors is the key to achieve good model quality.

Algorithm 11: FL-CS-DP: Private Compressive Sensing Federated Learning

Server:
  Initialize common model $\mathbf{w}_0$, η_G , ρ, $\mathbf{u}_t = 0$, $\mathbf{e}_t = 0$
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\mathbf{y}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{y}_t = \sum_{k=1}^{|\mathbb{K}|} \frac{\mathbf{y}_t^k}{|\mathbb{K}|}$: Averaging
    $\mathbf{u}_t = \rho \mathbf{u}_{t-1} + \mathbf{y}_t$: Momentum
    $\mathbf{e}_t = \eta_G \mathbf{u}_t + \mathbf{e}_{t-1}$: Error Feedback
    $\mathbf{s}_t = \mathcal{D}(\mathbf{e}_t, \mathbf{n})$: Reconstruction
    $\mathbf{e}_t = \mathbf{e}_t - \mathcal{C}(\mathbf{s}_t, m)$: Error accumulation
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{s}_t$: Update
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\mathbf{w}_{t-1}^k)$:
  $\mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
  $\Delta \mathbf{w}_t^k = \mathbf{w}_t^k - \mathbf{w}_{t-1}^k$
  $\mathbf{c}_t^k = \mathcal{C}(\Delta \mathbf{w}_t^k, m)$
  $\hat{\mathbf{c}}_t^k = \mathbf{c}_t^k / \max\left(1, \frac{||\mathbf{c}_t^k||_2}{S}\right)$
  Output: $\text{Enc}_{K_k}(\mathcal{G}(\hat{\mathbf{c}}_t^k, S\mathbf{I}\sigma/\sqrt{|\mathbb{K}|}))$

5.3.3 Robustness of FL-CS-DP Against non-adversarial Client Failures

If any client fails to add its noise share to the model update for any reason, the aggregate will not have sufficient amount of noise to guarantee differential privacy. A straightforward countermeasure is to increase the variance of the added noise so that even if l clients fail, the sum of $|\mathbb{K}| - l$ noise shares are still enough for differential privacy. In particular, each client k sends $\text{Enc}_{K_k}(\mathcal{G}(\hat{\mathbf{c}}_t^k, S\mathbf{I}\sigma/\sqrt{|\mathbb{K}| - l}))$ to the server for aggregation. Obviously, if less than r nodes fail, the aggregate will have larger noise than what is necessary for differential privacy.

5.3.4 Experimental Set-up

To evaluate our private scheme FL-CS-DP, we use exactly the same settings and datasets used to evaluate its non-private version FL-CS described above in Section 4.1.5. Note that, FL-STD-DP, FL-RND-DP and FL-FREQ-DP (see Alg 10,12,13 for more details) are the private extension of FL-STD, FL-RND and FL-FREQ, respectively.

The sensitivity S is selected during an initialization round for each scheme by taking the median value over N L_2-norm values. We also noticed that the sensitivity of FL-CS-DP, FL-RND-DP and FL-FREQ-DP are nearly equivalent for the same level of compression. For this reason, the same sensitivity value is used for each compressed scheme and for the same compression ratio. Table 8.4 and Table 8.5 show the selected clipping threshold (i.e., sensitivity S) for each dataset and according to each compression ratio.

5.3.5 Results

Table 5.3 represents the best accuracy over 200 rounds for each scheme on the Fashion-Mnist dataset. $Round$ corresponds to the round when the best accuracy is reached and $Cost$ is the average bandwidth consumption calculated as: $r \times n \times 32 \times Round \times C$, where 32 is the number of bits necessary to represent a float value, n is the uncompressed model size, $r = \frac{m}{n}$, m is the compressed

Algorithm 12: FL-RND-DP

Server:
  Initialize common model $\mathbf{w}_0$
  for $t = 1$ **to** T_{cl} **do**
    Generate a random seed ζ Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\mathbf{y}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1}, \zeta)$
    end
    $\mathbf{y}_t = \sum_k \frac{|D_k|}{\sum_j^N |D_j|} \Delta \mathbf{w}_t^k$
    $j = 0$
    for *each element* i *in* $\mathbf{G}$ **do**
      $\mathbf{w}_t[i] = \mathbf{w}_{t-1}[i] + \mathbf{y}_t[j]$
      $j = j + 1$
    end
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\mathbf{w}_{t-1}^k, \zeta)$**:**
  $\mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
  $\Delta \mathbf{w}_t^k = \mathbf{w}_t^k - \mathbf{w}_{t-1}^k$
  Generates a random set $\mathbf{G} = \{x \in \{1, \cdots, n\}\}$ of $\mathbf{m}$ random integer values such that $\mathbf{m} \leq \mathbf{n}$ based on the seed ζ
  $\hat{\Delta w_t^k}$ =Sample $\mathbf{m}$ elements from $\Delta \mathbf{w}_t^k$ by taking each element of $\mathbf{G}$ as a coordinate
  $\hat{\Delta w_t^k}' = \hat{\Delta w_t^k} / \max\left(1, \frac{||\hat{\Delta w_t^k}||_2}{S}\right)$
  Output: $\text{Enc}_{K_k}(\mathcal{G}(\hat{\Delta w_t^k}', S\mathbf{I}\sigma/\sqrt{|K|}))$

model size, C is the sampling probability of a client, and $Round$ is the round when we get the the best accuracy.

Table 5.4 represents the best balanced accuracy over 100 rounds for each scheme on the Medical dataset. AUROC (area under the receiver operating characteristic curve [Nar18]) corresponds to the AUROC value when the best balanced accuracy is reached, round is also the round when we get the best balanced accuracy, and finally, Cost is the average bandwidth consumption calculated as for the Fashion-MNIST dataset described above.

Surprisingly, for the smallest compression ratio 5%, FL-CS-DP performs as well as FL-RND-DP or FL-FREQ-DP in terms of accuracy and much better in terms of bandwidth consumption. Indeed, FL-CS-DP with a compression ratio of 5% reached 0.78 of accuracy on Fashion-MNIST, however, our baseline FL-RND-DP needs a compression ratio of 10% to reach a similar result (0.77) and 20% to have slightly better result (0.80). The same holds for the medical dataset, where FL-CS-DP reached 0.69 and 0.76 of balanced accuracy and AUROC, respectively. However, our other baseline FL-FREQ-DP needs a compression ratio of 20% to reach the same performance. FL-CS-DP performs better for a small compression ratio. Indeed, FL-CS-DP reaches 0.78, 0.73 and 0.66 for 5%, 10% and 20% of accuracy, respectively, on Fashion-MNIST. The accuracy degradation with DP can be explained by the fact increasing the compression ratio r also increases the sensitivity S which has a direct impact on the additive Gaussian noise as explained in Section 5.3.1. Indeed, the standard deviation of the normal distribution is $\sigma \times S$.

On both datasets, FL-STD-DP suffers from the noise due to the large sensitivity which is the largest one in Table 8.4 and Table 8.5 for a compression ratio of 1.0 (uncompressed model). Even for FL-RND and FL-FREQ, the gap between the non-private and the private version is larger when the compression ratio increases for both datasets. As the noise proportional to S and is added to every coordinate, the norm of the added noise increases with the model size n. This has negative impact on model convergence for a large n as discussed in [Zhu+20]. By decreasing n, compression helps reach better utility.

On Fashion-MNIST, FL-CS-DP with a compression ratio of 0.05 outperforms FL-STD-DP on both utility and bandwidth preservation. However, and even though FL-CS-DP reduces the bandwidth cost

Algorithm 13: FL-FREQ-DP

Server:
  Initialize common model $\mathbf{w}_0$
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\Delta \mathbf{y}_t^k = \mathbf{Client}_k(\mathbf{w}_{t-1})$
    end
    $\mathbf{y}_t = \sum_k \frac{|D_k|}{\sum_j^N |D_j|} \Delta \mathbf{w}_t^k$
    $\hat{\mathbf{y}}_t = \Phi^{-1}\mathbf{y}_t$: Transform to time domain
    $\mathbf{w}_t = \mathbf{w}_{t-1} + \hat{\mathbf{y}}_t$
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\mathbf{w}_{t-1}^k)$**:**
  $\mathbf{w}_t^k = \mathbf{SGD}(D_k, \mathbf{w}_{t-1}^k, T_{\text{gd}})$
  $\Delta \mathbf{w}_t^k = \mathbf{w}_t^k - \mathbf{w}_{t-1}^k$
  $\Delta \hat{w}_t^k = \Phi \Delta \mathbf{w}_t^k / \max\left(1, \frac{||\mathcal{C}(\Delta \hat{\mathbf{w}}_t^k, m)||_2}{S}\right)$
  Output: $\text{Enc}_{K_k}(\mathcal{G}(\Delta \hat{w}_t^k, S\mathbf{I}\sigma/\sqrt{|K|}))$

by 95% which is not negligible, they have both comparable accuracy on the medical data. It can be explained by the reduction of the noise due to the reduction of S and σ (see Table 8.5 and Table 8.4: Indeed, for FL-STD-DP, S*sigma=0.46 with the medical data, and it is 3.31 with Fashion-MNIST.) needed to reach an ϵ value of at most 1 after T_{cl} rounds. In order to validate our assumption, we have decided to run two more experiments: (1) We increased sigma from 1.49 to 5 (epsilon=0.39), which will result in a (balanced) accuracy of 0.65 and an AUROC value of 0.70 for FL-CS-DP (r=0.05), and only 0.63 (Balanced accuracy) and 0.68 (AUROC) for FL-STD-DP. (2) We have decided to run an experiment on another dataset, and we chose the well-known MNIST dataset. In this case the accuracy of FL-CS-DP (r=0.05, S=0.39) is equal to 0.93 and the accuracy of the FL-STD-DP (S=1.8) is 0.85. Once again, the results show the positive impact of compressive sensing on the quality of predictions as measured by AUROC, Balanced accuracy or Accuracy; under privacy-preserving settings.

There is a possible tradeoff between the privacy, communication cost, and utility. Indeed, having a small ϵ (better privacy) results in a reduction of the communication costs while it decreases accuracy. FL-STD-DP, for example, converges to the best accuracy (61%) after only 25 rounds with early stopping, which results in high privacy (ϵ=0.69) and low communication cost (only 22.18 Megabit). However, the accuracy degradation is more important (about 30% which is the worst accuracy degradation indicated in Table 5.3). Indeed, the large amount of added noise impacts the convergence of the model which can not achieve an accuracy larger than 61%.

Finally, we highlight a trade-off for FL-CS-DP. As mentioned above, FL-CS-DP performs better when the smallest compression ratio r is used, as the sensitivity for this level of compression is the smallest one. On the other hand, the compression ratio cannot be decreased arbitrarily as it will result in large reconstruction error. Therefore, one has to find the smallest compression ratio that is small enough to reduce the perturbation error but large enough to induce small reconstruction error. Finally, we note that increasing the number of local SGD iterations T_{gd} performed by each client reduces the sparsity of its updates. This can make the reconstruction of the aggregate at the server side less accurate or even impossible.

5.4 FL-TOP-DP: Private and Bandwidth-Efficient Federated Learning via Constraint

FL-TOP-DP is the private extension of FL-TOP described in Section 4.2.1. It provides the same guarantee as FL-SIGN-DP and FL-CS-DP and considers the same *Privacy model* (see Section 5.1 and Section 5.2.1 for more details). Therefore, it aims at preserving the privacy of each client instead of each record (client-level privacy). In what follows, we first describe the operation required to reach

Compression ratio (r)	Algorithms	*Performance*			
		Accuracy	*Round*	*Cost (Megabit)*	ϵ
0.05	FL-RND-DP	0.73	196	8.69	0.99
	FL-FREQ-DP	0.72	200	8,87	1
	FL-CS-DP	0.78	197	8.74	1
0.1	FL-RND-DP	0.77	199	17.65	1
	FL-FREQ-DP	0.76	200	17.74	1
	FL-CS-DP	0.73	101	8,96	0.84
0.2	FL-RND-DP	0.80	199	35.31	1
	FL-FREQ-DP	0.79	200	35,49	1
	FL-CS-DP	0.66	150	26,61	0.92
1.0	FL-STD-DP	0.61	25	22.18	0.69

Tab. 5.3: Summary of results on Fashion-MNIST dataset using the FL-CS-DP scheme.

Compression ratio (r)	Algorithms	*Performance*				
		Bal_Acc	*AUROC*	*Round*	*Cost(Megabit)*	ϵ
0.05	FL-RND-DP	0.60	0.69	100	4.78	1
	FL-FREQ-DP	0.65	0.72	100	4.78	1
	FL-CS-DP	0.69	0.76	100	4.78	1
0.1	FL-RND-DP	0.65	0.72	100	9.56	1
	FL-FREQ-DP	0.67	0.74	100	9.56	1
	FL-CS-DP	0.69	0.76	99	9.46	1
0.2	FL-RND-DP	0.67	0.74	99	18.92	1
	FL-FREQ-DP	0.69	0.76	100	19.11	1
	FL-CS-DP	0.68	0.74	64	12.23	0.92
1.0	FL-STD-DP	0.70	0.77	93	88.88	0.99

Tab. 5.4: Summary of results on Medical dataset using the FL-CS-DP scheme.

this guarantee and then we evaluate our private scheme on both: accuracy and bandwidth efficiency aspects.

5.4.1 Operation

FL-TOP-DP is described in Alg. 14 is very similar to FL-TOP except that each client adds Gaussian noise to its Top-K model updates to guarantee client-level DP, and applies secure aggregation allowing the server to learn only the aggregated (and noisy) model update. More specifically, each client first calculates its compressed model update $\Delta \mathbf{w}_t^k = \mathcal{C}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k)$ (in Line 25) which is then clipped (in Line 26) to obtain $\Delta \hat{\mathbf{w}}_t^k$ with L_2-norm at most S. After that, random noise $\mathbf{z}_k \sim \mathcal{G}(0, S\sigma\mathbf{I}/\sqrt{\mathbb{K}})$ is added to $\Delta \hat{\mathbf{w}}_t^k$ such that the sum $\sum_{k\in\mathbb{K}}(\Delta \hat{\mathbf{w}}_t^k + \mathbf{z}_k) = \sum_{k\in\mathbb{K}} \Delta \hat{\mathbf{w}}_t^k + \mathcal{G}(0, S\sigma\mathbf{I})$ as the sum of Gaussian random variables also follows Gaussian distribution[7] and then differential privacy is satisfied where ε and δ can be computed using the moments accountant described in Section 2.2. Recall that the Top-K coordinates in $\mathbb{T}$ are selected and distributed by the server, which is honest-but-curious by assumption.

However, as the noise is inversely proportional to $\sqrt{\mathbb{K}}$, $\mathbf{z}_k$ is likely to be small if $|\mathbb{K}|$ is too large. Therefore, the adversary accessing an individual update $\Delta \hat{\mathbf{w}}_t^k + \mathbf{z}_k$ can almost learn a non-noisy update since $\mathbf{z}_k$ is small. Hence, each client uses secure aggregation to encrypt its individual update before sending it to the server. Upon reception, the server sums the encrypted updates as:

$$\begin{aligned}\sum_{k\in\mathbb{K}} \mathbf{c}_t^k &= \sum_{k\in\mathbb{K}} \mathsf{Enc}_{K_k}(\Delta \hat{\mathbf{w}}_t^k + \mathbf{z}_k) = \sum_{k\in\mathbb{K}} \Delta \hat{\mathbf{w}}_t^k + \sum_{k\in\mathbb{K}} \mathbf{z}_k \\ &= \sum_{k\in\mathbb{K}} \Delta \hat{\mathbf{w}}_t^k + \mathcal{G}(0, S\sigma\mathbf{I}) \end{aligned} \tag{5.7}$$

[7] More precisely, $\sum_i \mathcal{G}(\nu_i, \xi_i) = \mathcal{G}(\sum_i \nu_i, \sqrt{\sum_i \xi_i^2})$

where $\mathsf{Enc}_{K_k}(\Delta\hat{\mathbf{w}}_t^k + \mathbf{z}_k) = \Delta\hat{\mathbf{w}}_t^k + \mathbf{z}_k + \mathbf{K}_k \mod p$ and $\sum_k \mathbf{K}_k = 0$ (see [ÁC11; Bon+16] for details). Here the modulo is taken element-wise and $p = 2^{\lceil \log_2(\max_k ||\Delta\hat{\mathbf{w}}_t^k + \mathbf{z}_k||_\infty |\mathbb{K}|)\rceil}$. Let $\gamma_t^k = 1/\max\left(1, \frac{||\Delta \mathbf{w}_t^k||_2}{S}\right)$. Then,

$$\begin{aligned}\sum_{k\in\mathbb{K}} \Delta\hat{\mathbf{w}}_t^k &= \sum_{k\in\mathbb{K}} \gamma_t^k \Delta\mathbf{w}_t^k = \sum_{k\in\mathbb{K}} \gamma_t^k \mathcal{C}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k, \mathbb{T}) \\ &= \mathcal{C}(\sum_{k\in\mathbb{K}} \gamma_t^k (\mathbf{w}_t^k - \mathbf{w}_{t-1}^k), \mathbb{T}) \end{aligned} \tag{5.8}$$

where the last equality comes from the linearity of the compression operation. Indeed, recall that each client selects the values of the *same* Top-K coordinates from $\mathbb{T}$. Plugging Eq. (5.8) into Eq. (5.7). we get that

$$\sum_{k\in\mathbb{K}} \mathbf{c}_t^k = \mathcal{C}(\sum_{k\in\mathbb{K}} \gamma_t^k (\mathbf{w}_t^k - \mathbf{w}_{t-1}^k), \mathbb{T}) + \mathcal{G}(0, S\sigma\mathbf{I})$$

5.4.2 Privacy of FL-TOP-DP

Privacy analysis: The server can only access the noisy aggregate which is sufficiently perturbed to ensure differential privacy; any client-specific information that could be inferred from the noisy aggregate is tracked and quantified by the moments accountant, described in Section 2.2, as follows.

Let $\eta_0(x|\xi) = \mathsf{pdf}_{\mathcal{G}(0,\xi)}(x)$ and $\eta_1(x|\xi) = (1-C)\mathsf{pdf}_{\mathcal{G}(0,\xi)}(x) + C\mathsf{pdf}_{\mathcal{G}(1,\xi)}(x)$ where C is the sampling probability of a single client in a single round. Let $\alpha(\lambda|C) = \log\max(E_1(\lambda,\xi,C), E_2(\lambda,\xi,C))$ where $E_1(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_0(x|\xi,C)\cdot\left(\frac{\eta_0(x|\xi,C)}{\eta_1(x|\xi,C)}\right)^\lambda dx$ and $E_2(\lambda,\xi,C) = \int_{\mathbb{R}} \eta_1(x|\xi,C)\cdot\left(\frac{\eta_1(x|\xi,C)}{\eta_0(x|\xi,C)}\right)^\lambda dx$.

Theorem 5 (Privacy of FL-TOP-DP)**.** *FL-TOP-DP is* $(\min_\lambda(T_{\mathsf{cl}}\cdot\alpha(\lambda|C) - \log\delta)/\lambda, \delta)$*-DP.*

Given a fixed value of δ, ε is computed numerically as in [Aba+16; MTZ19].

Algorithm 14: FL-TOP-DP: Federated Learning

Server:
  Initialize common model w_0
  Select set $\mathbb{T}$ of Top-K updated weights' coordinates via public dataset
  for $t = 1$ **to** T_{cl} **do**
    Select $\mathbb{K}$ clients uniformly at random
    for *each client* k *in* $\mathbb{K}$ **do**
      $\mathbf{c}_t^k = \mathbf{Client}_k(\mathcal{C}(\mathbf{w}_{t-1}, \mathbb{T}))$
    end
    $\mathbf{w}_t = \mathbf{w}_0$
    $j = 1$
    for *each coordinate* i *in* $\mathbb{T}$ **do**
      $\mathbf{w}_t[i] = \mathbf{w}_{t-1}[i] + \sum_k \frac{\mathbf{c}_t^k[j]}{|\mathbb{K}|}$
      $j = j + 1$
    end
  end
  Output: Global model $\mathbf{w}_t$

$\mathbf{Client}_k(\hat{\mathbf{w}}_{t-1}^k)$:
  $\mathbf{w}_{t-1}^k = \mathbf{w}_0$
  $j = 1$
  for *each coordinate* i *in* $\mathbb{T}$ **do**
    $\mathbf{w}_{t-1}^k[i] = \hat{\mathbf{w}}_{t-1}^k[j]$
    $j = j + 1$
  end
  $\mathbf{w}_t^k = \mathbf{Top_k SGD}(D_k, \mathbf{w}_{t-1}^k, \mathbf{w}_0, T_{\mathsf{gd}}, \mathbb{T})$
  $\Delta\mathbf{w}_t^k = \mathcal{C}(\mathbf{w}_t^k - \mathbf{w}_{t-1}^k, \mathbb{T})$
  $\Delta\hat{\mathbf{w}}_t^k = \Delta\mathbf{w}_t^k / \max\left(1, \frac{||\Delta\mathbf{w}_t^k||_2}{S}\right)$
  Output: $\mathsf{Enc}_{K_k}(\mathcal{G}(\Delta\hat{\mathbf{w}}_t^k, S\mathbf{I}\sigma/\sqrt{|\mathbb{K}|}))$

5.4.3 Remarks

The magnitude of the added Gaussian noise is proportional to the clipping threshold S, which is in turn calibrated to the norm of the model update. However, the norm of the model update increases if the model size increases [Zhu+20], and hence S should be chosen sufficiently large to guarantee fast convergence with large accuracy. On the other hand, too large S also increases the perturbation error caused by the added noise.

FL-TOP aims to diminish this perturbation error by reducing S via compression which also increases the L_2-norm of the compressed update vector. This is illustrated in Figure 5.1, which shows that the norm of the Top-K coordinates with FL-TOP tend to be larger than with FL-STD (i.e., when all coordinates get updated not only the Top-K). Therefore, besides decreasing the magnitude of the added noise, FL-TOP also decreases the relative error on the retained parameters. These together decrease the perturbation error caused by the added noise.

Notice that there exist other alternatives to identify the Top-K coordinates in a privacy-preserving manner than using a public dataset. For example, every client can select the Top-K parameters with the largest magnitude during the first rounds locally, and send them to the server for aggregation. More specifically, each client creates a parameter vector with size n, where the Top-K coordinates are set to 1 while the rest are kept 0. Then, these binary vectors are noised and aggregated by the server like in Section 5.4.1. In the rest of the training, all participants exchange only the updates and weights of the these Top-K parameters like in FL-TOP. However, aside from consuming more privacy budget, this approach also has lower accuracy than our proposal according to our tests. Moreover, it has larger communication cost in the initialization phase when the Top-K parameters are identified and the whole binarized parameter vector is sent for aggregation.

5.4.4 Robustness of FL-TOP-DP Against non-adversarial Client Failures

If any client fails to add its noise share to the model update for any reason, the aggregate will not have sufficient amount of noise to guarantee differential privacy. A straightforward countermeasure is to increase the variance of the added noise so that even if l clients fail, the sum of $|\mathbb{K}| - l$ noise shares are still enough for differential privacy. In particular, each client k sends $\mathsf{Enc}_{K_k}(\mathcal{G}(\Delta\hat{\mathbf{w}}_t^k, S\mathbf{I}\sigma/\sqrt{|\mathbb{K}|}))$ to the server for aggregation. Obviously, if less than r nodes fail, the aggregate will have larger noise than what is necessary for differential privacy.

5.4.5 Experimental Set-up

The goal of this section is to evaluate the performance of our proposed scheme FL-TOP-DP on a benchmark dataset and a realistic in-hospital mortality prediction scenario. We aim at evaluating their performance with different levels of compression and comparing them with the performance of the following learning protocols: FL-STD-DP,FL-BASIC-DP, FL-CS-DP and FL-SIGN-DP which are the private extensions of FL-STD, FL-BASIC, FL-CS and FL-SIGN introduced in Section 2.1, Section 4.2.2, Section 4.1.4 and Section 4.1.1[8].

Note that all compression operators in the baselines are linear (just like FL-TOP-DP), and hence they can also be used with secure aggregation. Similarly to FL-TOP-DP, the private extensions (i.e., FL-STD-DP, FL-BASIC-DP, FL-CS-DP and FL-SIGN-DP) also clip and then noise the compressed updates. However, with FL-SIGN-DP we use the Discrete Gaussian Mechanism defined in Section 5.2.2 instead of the continuous.

Similarly to Section 4.2.2, we evaluate the above learning algorithms on the well-known Fashion-MNIST dataset and on the Premier Healthcare Database. More details can be found in Section 8.3. We also fix the Top-K weights before starting the federated learning process using public data: For Fashion-MNIST we use MNIST and for the medical dataset we use a batch from the same dataset (as we did in Section 4.2.2).

In order to select the clipping threshold S, the server executes a single training round locally, which is composed of T_{gd} SGD iterations starting from the model parameters $\mathbf{w}_0$, using the batch from the public data. The clipping threshold S is set to the L_2-norm of the Top-K weight update obtained

[8] More baselines are considered but we have decided to present only those which return the best results. All other results can be found in Tables 8.11,8.12,8.13.

for this single training round. For FL-BASIC-DP, the same steps are repeated for 100 times, where a new random set of trainable weights with size K are selected each time, which yields 100 L_2-norm values. S is set to the median of these L_2-norm values. We think that this approach is more fair, because the set of trainable weights is re-selected at each round in FL-BASIC-DP. The computed values of S can be found in Table 8.9 and Table 8.10 for Fashion-MNIST and Medical dataset, respectively. More information about the model architecture and the hyper-parameter selection can be found in Section 8.3 and Table 8.8 in the Appendix.

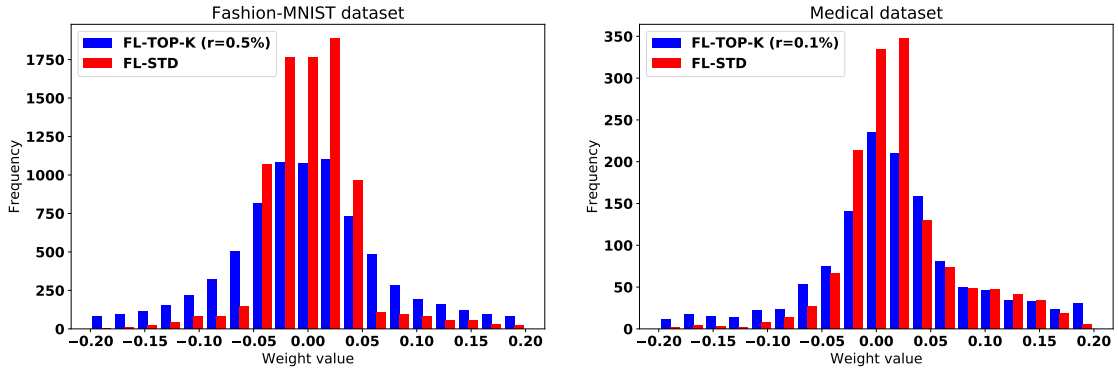

Fig. 5.1: Distributions of the Top-K weight values (after convergence) for both FL-TOP and FL-STD schemes with the Fashion-MNIST dataset (left) and the medical dataset (right).

r	Algorithms	*Performance*				
		Accuracy	*Round*	*Downstream Cost* (Kilobyte)	*Upstream Cost* (Kilobyte)	ϵ
0.5%	FL-BASIC-DP	0.59	200	22178.27	110.88	1
	FL-CS-DP	0.53	200	22178.27	110.88	1
	FL-TOP-DP	**0.81**	200	**110.88**	**110.88**	**1**
5%	FL-BASIC-DP	0.76	195	21623.81	1081.18	0.99
	FL-CS-DP	0.78	160	17742.61	887.13	0.94
	FL-TOP-DP	**0.81**	152	**842.77**	**842.77**	**0.92**
10%	FL-BASIC-DP	0.79	189	20958.46	2095.85	0.98
	FL-CS-DP	0.72	167	18518.85	1851.89	0.95
	FL-TOP-DP	**0.80**	157	**1740.99**	**1740.99**	**0.93**
100%	FL-STD-DP	0.56	60	6653.48	6653.48	0.76
	FL-SIGN-DP	0.63	187	20736.68	14256,47	0.98

Tab. 5.5: Summary of results on Fashion-MNIST dataset using the FL-TOP-DP scheme.

5.4.6 Results

Figure 5.1 displays the distribution of the Top-K updated weights for FL-TOP and FL-STD at the end of the training. We select the weights when each scheme reached the best accuracy over 200 and best balanced accuracy[9] over 100 rounds for fashion-MNIST and the medical dataset, respectively. We choose the smallest compression ratio r that leads to the best accuracy for the FL-TOP-DP scheme. Table 5.5 shows that FL-TOP-DP reaches the best accuracy, 0.81, when $r = 0.5\%$ on fashion-MNIST and reaches the best accuracy, 0.69, when $r = 0.1\%$ on the medical dataset. Both figures validate the intuition that by constraining the model to update only a small set K of the total weights, these Top-K become more important and reach larger values. This result is important when differential privacy is used as it leads to larger value-to-noise level and therefore better performance.

Table 5.5 represents the best accuracy over 200 rounds for each scheme on the Fashion-MNIST dataset. $Round$ corresponds to the round when the best accuracy is reached and $Cost$ is the average bandwidth consumption calculated as: $r \times n \times 32 \times Round \times C$, where 32 is the number of bits

[9] See Appendix 8.4.2 for more details.

r	Algorithms	*Performance*					
		Bal_Acc	*AUROC*	*Round*	*Downstream Cost* (Kilobyte)	*Upstream Cost* (Kilobyte)	ϵ
0.1%	FL-BASIC-DP	0.50	0.49	100	11948.91	11.94	1
	FL-CS-DP	0.51	0.51	99	11829.42	11.82	1
	FL-TOP-DP	**0.69**	**0.76**	85	**10.15**	**10.15**	**0.97**
5%	FL-BASIC-DP	0.69	0.76	100	11948.91	597.45	1
	FL-CS-DP	0.69	0.76	100	11948.91	597.45	1
	FL-TOP-DP	**0.68**	**0.75**	23	**137.41**	**137.41**	**0.79**
10%	FL-BASIC-DP	0.69	0.76	99	11829.42	1182.94	1
	FL-CS-DP	0.69	0.76	96	11470.95	1147.09	0.99
	FL-TOP-DP	**0.68**	**0.74**	23	**274.82**	**274.82**	**0.79**
100%	FL-STD-DP	0.66	0.72	62	7408.32	7408.32	0.91
	FL-SIGN-DP	0.63	0.68	97	11590,44	7968,43	1.0

Tab. 5.6: Summary of results on Medical dataset using the FL-TOP-DP scheme.

necessary to represent a float value, n is the uncompressed model size, $r = \frac{|\mathbb{T}|}{n}$, $|\mathbb{T}|$ is the compressed model size, C is the sampling probability of a client, and $Round$ is the round when we get the the best accuracy.

Table 5.6 represents the best balanced accuracy over 100 rounds for each scheme on the Medical dataset. $AUROC$ (area under the receiver operating characteristic curve - see Appendix 8.4.2) corresponds to the $AUROC$ value when the best balanced accuracy is reached. Note that FL-SIGN-DP needs 22 bits to encode each parameter as it is based on the Discrete Gaussian Mechanism (more details can be found in Section 5.2.4).

The results also show that not only our privacy-preserving solution FL-TOP-DP provides strong privacy guarantee (with ϵ values smaller than 1) but also that it outperforms the other schemes in terms of accuracy and bandwidth, for both datasets.

For example, with Fashion-MNIST, our scheme achieves an accuracy of 0.81 when $r = 0.5\%$ while the baseline scheme, FL-BASIC-DP, achieves an accuracy of 0.79 when $r = 10\%$ and requires 189 times more downstream bandwidth and 18 times more upstream bandwidth. Similarly, FL-SIGN-DP is 22% less accurate than FL-TOP-DP and it requires 17 times more upstream bandwidth and 25 times more downstream bandwidth.

With the medical dataset, FL-TOP-DP reaches the best balanced accuracy 0.69 and best AUROC 0.76 for a compression ratio of $r = 0.1\%$ while FL-BASIC-DP and FL-CS-DP achieve the same performance at $r = 5\%$. Note that FL-STD-DP performs very poorly as noise has to be added to the all weights of the model and the sensitivity is large (see Table 5.5 and Table 5.6). Finally, FL-SIGN-DP is 9% and 11% less accurate than FL-TOP-DP on the Balanced accuracy and AUROC, respectively, while it requires 1142 times more upstream bandwidth and 785 times more downstream bandwidth.

5.5 Conclusion

Differential privacy generally consists of adding noise sampled from a continuous Gaussian distribution to the updates. However, if we add continuous Gaussian noise to the quantized updates of FL-SIGN in order to obtain its private extension called FL-SIGN-DP, we loose the bandwidth efficiency property. We therefore proposed to sample noise from a discrete Gaussian distribution instead. As a result, FL-SIGN-DP is 30-40% more bandwidth efficient than FL-STD and FL-STD-DP.

We also have shown that the utility of the private model can be improved by reducing its sensitivity via compression. Similarly, it can be improved by increasing the value-to-noise either by taking the first frequencies of the discrete cosine transform of the update (FL-CS-DP) or by constraining the model to only update some weights (FL-TOP-DP). Finally, reconstructing the approximated gradients in FL-CS-DP is an instance of Basis Pursuit Denoising (or LASSO), which can be solved with efficient solvers that provide large accuracy despite the added Gaussian noise.

6 Security Analysis

In this Chapter, we first evaluate the robustness of FL-SIGN, FL-CS and FL-TOP against several state-of-the-art security attacks. More specifically, we investigate the robustness against pollution attacks which includes targeted (backdoors attacks) and untargeted attacks. We noticed that FL-SIGN is particularly robust to such attacks compared to the vanilla federated learning FL-STD, but also to our other compressed schemes, namely FL-CS and FL-TOP. Indeed, we have shown that FL-SIGN prevents the adversary to scale up its update to increase its impact on the global model. Due to the quantized nature of FL-SIGN, the adversary can be easily detected if it sends a value larger than $+1$ or smaller than -1. Also, the majority vote at the server side also allows the server to be more robust against the attacks.

In the second part of this Chapter, we investigate if FL-SIGN-DP is as robust as its non-private variant FL-SIGN. However, we noticed that FL-SIGN-DP is not robust. Seemingly, there is a possible trade-off between differential privacy and robustness against security attacks.

6.1 Security Model

Adversarial model: In this work, we assume that the adversary controls a certain fraction of the participating entities/clients at each round of the training, which means it can access and modify these clients' training data as well as all parameters of their local model. We, however, assume that *the server is honest* (i.e., it does not manipulate the aggregate nor the update vector sent by any client). The set of all malicious nodes is denoted by $\mathbb{M}$.

We consider two types of adversary. The first one aims at degrading the overall model performance (i.e., increase the average misclassification rate). The second one aims at causing targeted misclassification on some particular classes of samples by injecting backdoors into the model during the training phase. These adversaries are *active* in the sense that they may not follow the learning protocol faithfully.

Next, we detail the attacks considered.

6.1.1 Overall Model Degradation Attacks (Untargeted Attacks)

6.1.1.1 Random Update Attack

In this attack, malicious clients, whose numbers might vary as shown later, use random updates. More specifically, instead of the true model update $\Delta \mathbf{w}_t^k$, each malicious client k generates a random update $\Delta \hat{\mathbf{w}}_t^k$ in all time slots t [BEGS17], where $\Delta \hat{\mathbf{w}}_t^k$ is drawn from an isotropic Gaussian distribution $\mathcal{G}(0, \sigma_{\mathsf{Adv}} \mathbf{I})$ with mean zero and variance σ_{Adv}^2. Each malicious party selects the noise independently (i.e., they do not collude).

6.1.1.2 Gradient Ascent Attack

In this attack, malicious clients aim at maximizing the loss by performing gradient *ascent* instead of descent on their own training data [NSH19]. In particular, *every* malicious client $k \in \mathbb{M}$ updates the model parameters locally as $\mathbf{w}_\ell^k = \mathbf{w}_{\ell-1}^k + \eta_{\mathsf{Adv}} \nabla f(\cup_{k \in \mathbb{M}} D_k; \mathbf{w})$, where η_{Adv} is set in order to suppress the updates of honest clients and to maximize the impact of their own update on the common model. Notice that this attack assumes *colluding malicious clients* (i.e., every malicious client sends exactly the same update computed on the union of their training data). This attack attempts to maximize the *average* misclassification rate of the common model, and is more effective if the number of malicious parties is large, or the training data of the malicious and benign nodes come from similar distributions.

We note that Gradient Ascent Attack is equivalent to the Sign Inversion Attack for FL-SIGN, described in [BZAA18], if $T_{\mathsf{gd}} = 1$ (i.e., each client computes its update using a single mini-batch in every round). In Sign Inversion Attack, all malicious clients faithfully compute the sign of their model update, but then send the *inverted* signs to the server for aggregation.

6.1.2 Backdoor Attacks (Targeted Attacks)

The goal of these attacks is to selectively degrade the accuracy of the common model with respect to only a few tasks. As opposed to the overall model degradation attacks, they generate *targeted* misclassification while preserving the model convergence as well as a high average prediction accuracy except, of course, for the targeted tasks, called backdoor classes.

We distinguish two types of backdoors: In-backdoors and Out-backdoors.

- *In-backdoor Attacks:* In-backdoor attacks [BCMC19] are created for a class of samples that exists in the training data of some parties. Specifically, for some training samples $D_{aux} \subseteq D_k$, each adversary uses output labels that are different from their true labels. Let y' denote the adversarially chosen label for a training sample $(x, y) \in D_{aux}$, and D'_{aux} denotes the set of all relabelled samples (i.e., $(x, y') \in D'_{aux}$). The new objective is to minimize the loss $f((D_k \setminus D_{aux}) \cup D'_{aux}; \mathbf{w})$.

- *Out-backdoor Attacks:* As opposed to in-backdoors, *out-backdoors* are created from samples that do *not* exist in the training data of any honest clients and are relabelled to have a class that *does* exist in their training data. Specifically, let L denote the set of labels that exist in $D = \cup_k D_k$. The adversary creates D''_{aux} such that, for each $(x, y) \in D''_{aux}$, $x \notin D$ and $y \in L$. The new objective is to minimize the loss $f(D_k \cup D''_{aux}; \mathbf{w})$.

To illustrate the difference between in- and out-backdoors, consider a model which recognizes dogs and rabbits in the input photos. If the adversary relabels all photos of dogs as 'rabbit' in its training data, then it is an in-backdoor attack. However, if the adversary adds new photos of frogs to its training data and relabels them as 'dog', then this is an out-backdoor attack.

Out-backdoors are more difficult to detect than in-backdoors as they can come from a much larger set of samples, which are potentially unknown to the protocol participants. Hence, out-backdoors are especially severe in security-related applications such as in access control.

As per [BCMC19], the adversary also uses explicit boosting to outbalance the combined effect of benign model updates. For both in- and out-backdoors, the adversary boosts $\Delta \mathbf{w}^t_{\mathsf{Adv}}$ at time t by sending $\eta_{\mathsf{Adv}} \Delta \mathbf{w}^t_{\mathsf{Adv}}$ ($\eta_{\mathsf{Adv}} > 1$) in order to suppress the model updates of benign parties. Importantly, η_{Adv} should be large enough in order to achieve misclassification of the backdoor class but also small enough to ensure the convergence of the common model and hence hide the attack.

6.2 Security Analysis of FL-SIGN, FL-CS and FL-TOP

In this section, we evaluate the robustness of FL-SIGN, FL-CS and FL-TOP against the security attacks presented previously.

For the Overall Model Degradation attacks, different percentage of malicious nodes are considered, the MNIST and IMDB datasets were used, and the same experimental setting as defined in Table 8.1 is used. The boosting parameter η_{Adv} of the Gradient Ascent Attack is set to 10 with MNIST dataset and 20 with the IMDB dataset (we use the boosting only with FL-STD, FL-CS and FL-TOP[1]). We also do not need to use boosting for the Random Update Attack as $\sigma_{\mathsf{Adv}} = 200$ generates large noise which prevents the model convergence.

For the Backdoor attacks, the MNIST and CIFAR datasets were used and the experimental setting is shown in Table 8.3. As backdoor attacks, which aim at modifying the prediction of one particular label while maintaining the global accuracy, are more difficult to perform on binary classifiers, we switched to the CIFAR dataset with a multiclass classifier. Furthermore, similarly to [BCMC19], we reduce the total number of clients N from 1000 to 10, we use different percentages of malicious nodes: 10%, 20% and 40%, and all clients report their updates to the server at each round (i.e. $C = 1.0$). The malicious nodes collude by sharing their data for the training and by sending the same update to the server.

6.2.1 Overall Model Degradation Attacks

6.2.1.1 Random Update Attack

Table 6.1 and 6.2 depict the best accuracy of the global model over 100 rounds according to the fraction of malicious nodes in set $\mathbb{K}$. The results show that FL-SIGN is robust against the random update attack even if 20% of all nodes are malicious, while FL-STD, FL-CS and FL-TOP fail to converge even if 1% of all nodes are malicious. Indeed, with 20% of malicious nodes, FL-SIGN reaches an accuracy of 98% and 86% for the MNIST and IMDB datasets, respectively. On the contrary, FL-STD, FL-CS and FL-TOP fail to converge even with one malicious node at each round. In fact, as we show in Appendix 8.1.4, FL-SIGN's convergence rate is $O\left(\frac{1}{(1-\alpha)\sqrt{CNT_{\mathsf{cl}}}}\right)$, where α denotes the fraction of malicious clients. This is in contrast to the sign inversion attack detailed in [BZAA18], which has a convergence rate of $O\left(\frac{1}{(1-2\alpha)\sqrt{CNT_{\mathsf{cl}}}}\right)$, that is, convergence is only possible if less than half of the nodes are malicious.

Tab. 6.1: Random update attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the MNIST dataset depending on the fraction of malicious nodes. $\sigma_{\mathsf{Adv}} = 200$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge.

	10%	20%	40%	60%
FL-STD	-	-	-	-
FL-CS	-	-	-	-
FL-TOP	-	-	-	-
FL-SIGN	0.98	0.98	0.94	-

[1] Due to quantization, boosting is not possible on FL-SIGN.

Tab. 6.2: Random update attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the IMDB dataset depending on the fraction of malicious nodes. $\sigma_{\text{Adv}} = 200$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge.

	10%	20%	40%	60%
FL-STD	-	-	-	-
FL-CS	-	-	-	-
FL-TOP	-	-	-	-
FL-SIGN	0.86	0.86	0.54	-

6.2.1.2 Gradient Ascent Attack

Table 6.3 and 6.4 show the best accuracy of the global model over 100 rounds when the adversary aims to degrade the average model performance by performing gradient ascent on its own training data. With the MNIST dataset (in Table 6.3), FL-SIGN reaches an accuracy of 98% and 79% for 20% and 40% of malicious nodes, respectively, while the other schemes do not converge even if only 10% of the nodes are malicious. For IMDB dataset, FL-SIGN reaches an accuracy of 86% and 72% for 10% and 20% of malicious nodes, respectively, while FL-STD, FL-CS and FL-TOP fail to converge with only 10% of malicious nodes. Indeed, they do not converge even if we have only one malicious node. The reason for this difference is that malicious nodes can scale up their update with η_{Adv} and hence boost its effect on the global model. However, such adversarial boosting does not work with FL-SIGN as the trusted server accepts only the values -1 and $+1$ in the update vectors. Therefore, a single malicious client does not have larger impact on the global model than any other honest client. To boost its impact, the adversary can only increase the number of the malicious clients, as shown by the experimental results. Since Gradient Ascent is equivalent to Sign Inversion Attack if $T_{\text{gd}} = 1$, the convergence rate of Gradient Ascent in this restricted scenario is $O\left(\frac{1}{(1-2\alpha)\sqrt{CNT_{\text{cl}}}}\right)$ as shown in [BZAA18].

	10%	20%	40%	60%
FL-STD	-	-	-	-
FL-CS	-	-	-	-
FL-TOP	-	-	-	-
FL-SIGN	0.98	0.98	0.79	-

Tab. 6.3: Gradient ascent attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the MNIST dataset. $\eta_{\text{Adv}} = 10$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge.

	10%	20%	40%	60%
FL-STD	-	-	-	-
FL-CS	-	-	-	-
FL-TOP	-	-	-	-
FL-SIGN	0.86	0.72	0.52	-

Tab. 6.4: Gradient ascent attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the IMDB dataset. $\eta_{\text{Adv}} = 20$. The table represents the best accuracy over 100 rounds. "-" means that the algorithm does not converge.

6.2.2 Backdoor Attacks

6.2.2.1 In-backdoor Attack

Figure 6.1, 6.2 and Table 6.5, 6.6 show the effect of in-backdoor attacks on the MNIST and CIFAR datasets, respectively. In all experiments, there are ten clients, out of which different fraction of malicious nodes are considered. Figure 6.1 depicts the accuracy of the global model for MNIST, when the adversary relabels every image of digit '5' to '7' in its local dataset. The red plots show the accuracy of the global models, while the green ones display the model accuracy only for the images with label '5' (i.e., accuracy on the backdoor class). The results show that FL-SIGN is robust as both global model accuracy and model accuracy on the specific in-backdoor class (digit 5) reach 99% by the end of the training.

By contrast, with FL-STD, while the accuracy of the global model converges slowly to 99%, the accuracy of the attacked model oscillates. Similar behaviour can be observed in Figure 6.2 which plots the accuracy on CIFAR dataset, where images of airplanes are re-labelled to 'ship' in the adversary's training data. In these experiments, FL-STD fails to converge on the backdoor class, and its accuracy on CIFAR never exceeds 55%.

The oscillation of accuracy with FL-STD can be explained by the nature of gradient descent and in particular backpropagation: when the malicious client injects the backdoor, it scales its update with η_{Adv}. In the following round, honest clients scale up their gradients on the backdoor samples (i.e., images of digit 5 in MNIST and images of airplanes in CIFAR) in order to "fix" the classification error on the backdoor class. In the next round, when the model is "fixed" (i.e., digit '5' is correctly predicted as '5' again), the adversary's gradients are increased again in order to re-inject the backdoor. This process repeats till the end of the training. By contrast, and similarly to the overall model degradation attacks, a malicious client cannot scale up its update in FL-SIGN as the update vectors must take value from $\{-1, 1\}^n$.

Table 6.5 shows the accuracy of the model on digit class 5 (in-backdoor class) when we consider different percentage of malicious nodes (values are chosen based on the best model accuracy over 40 rounds). The global accuracy of the model over all the classes is 99% and the accuracy on class '5' is 99% independently of the number of malicious nodes and regardless whether FL-STD or FL-SIGN is used. However, the accuracy on class '5' is more impacted when we use FL-CS or FL-TOP. Indeed, with 40% of malicious nodes, the accuracy on class '5' is 74% and 83% for FL-CS and FL-TOP, respectively.

As in the previous table, Table 6.6 shows the accuracy of the model on airplane class (in-backdoor class) when we consider different number of malicious nodes (values are chosen based on the best global accuracy over 100 rounds). FL-SIGN with 20% of malicious nodes reaches a global accuracy of 84%, and an accuracy of 76% on the in-backdoor class. However, FL-STD with the same amount of malicious nodes reaches 80% of global accuracy and 0% for the airplane class. About other schemes, FL-CS reaches a global accuracy of 80% and an accuracy of 71% on the 'airplane' class, whereas, FL-TOP reaches a global accuracy of 81% and an accuracy of 35% on the in-backdoor class. The results validate the robustness of FL-SIGN over the other schemes.

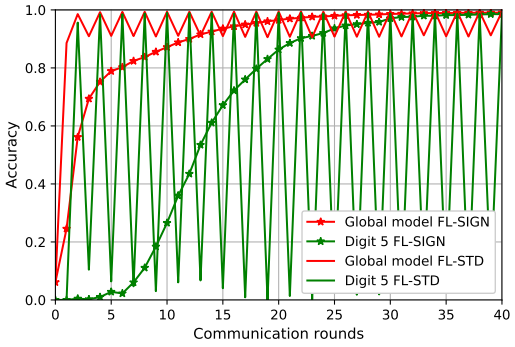

Fig. 6.1: In-backdoor attack on FL-SIGN and FL-STD with the MNIST dataset, $\eta_{\text{Adv}} = 7$. The figure displays the global accuracy convergence and the accuracy of the label "5" which is under attack. 10% of the nodes are malicious.

6.2.2.2 Out-backdoor Attack

The main goal of the out-backdoor attack is to introduce fake information during the training by relabeling a sample, whose true label is not a valid output of the global model. We experimented this attack on MNIST by first excluding all samples with digit '0' in all clients' training datasets. We then choose different fraction of malicious clients and relabeled the samples with '0' to '1'. Similarly, the attack is also implemented using the CIFAR dataset by removing all airplanes from the clients' training data and relabelling all images of an airplane as 'ship' in the malicious clients' datasets [2]. Note that

[2] we also removed all birds and trucks from the training data, in order to limit the bias between classes.

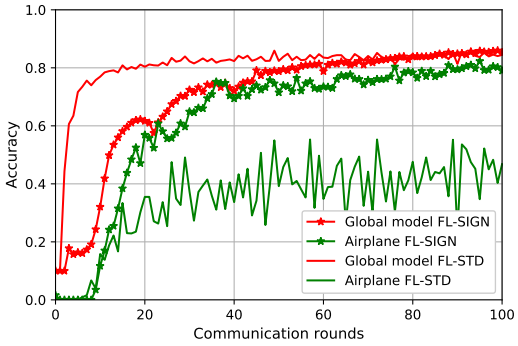

Fig. 6.2: In-backdoor attack on FL-SIGN and FL-STD with CIFAR dataset, $\eta_{\text{Adv}} = 7$. The figure displays the global accuracy convergence and the accuracy of the label "airplane" which is under attack. 10% of the nodes are malicious.

		10%	20%	40%
FL-SIGN	Model accuracy	0.99	0.99	0.99
	Accuracy on digit class '5'	0.99	0.99	0.99
FL-STD	Model accuracy	0.99	0.99	0.99
	Accuracy on digit class '5'	0.99	0.99	0.99
FL-CS	Model accuracy	0.99	0.99	0.97
	Accuracy on digit class '5'	0.98	0.96	0.74
FL-TOP	Model accuracy	0.99	0.99	0.97
	Accuracy on digit class '5'	0.98	0.95	0.83

Tab. 6.5: In-backdoor attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the MNIST dataset, η_{Adv} is set to 7, 3, 1 for 10%, 20%, 40% respectively. The table depicts the global accuracy convergence and the accuracy of the label "5" which is under attack. We choose the compression ratio r for FL-CS and FL-TOP such that each scheme reaches the same accuracy as FL-STD on the same settings but without the attack. Therefore, r is set to 5% and 0.5%, respectively. Note also, that we do not use boosting with FL-CS to not impact the sparsity of the model's update.

		10%	20%	40%
FL-SIGN	Model accuracy	0.86	0.84	0.82
	Accuracy on airplane class	0.80	0.76	0.57
FL-STD	Model accuracy	0.86	0.80	0.81
	Accuracy on airplane class	0.55	0	0
FL-CS	Model accuracy	0.81	0.80	0.74
	Accuracy on airplane class	0.81	0.71	0.30
FL-TOP	Model accuracy	0.82	0.81	0.80
	Accuracy on airplane class	0.42	0.35	0.18

Tab. 6.6: In-backdoor attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the CIFAR dataset, η_{Adv} is set to 7, 4, 2 for 10%, 20%, 40% respectively. The table depicts the global accuracy convergence and the accuracy of the label "airplane" which is under attack. We choose the compression ratio r for FL-CS and FL-TOP such that each scheme reaches the same accuracy as FL-STD on the same settings but without the attack. Therefore, r is set to 60% and 20%, respectively. Note also, that we do not use boosting with FL-CS to not impact the sparsity of the model's update.

since only malicious clients have samples from the backdoor class, the detection of this attack is quite challenging.

Tables 6.7 and 6.8 display the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class to the targeted class (attack accuracy) for MNIST and CIFAR, respectively (values are chosen based on the best model accuracy over 100 rounds with MNIST and 300 rounds with CIFAR). We consider different fraction of malicious nodes. The results show that the model accuracy is similar for both datasets and schemes, but FL-SIGN is much more robust against the attacks than other schemes. In fact, with 10% of malicious nodes, the attack accuracy on the MNIST dataset is very low for FL-SIGN (19%) whereas it is quite large for FL-STD (92%). We obtained similar

pattern with the CIFAR dataset although the accuracy difference is less significant (66% versus 72%). This can be explained by the inherent bias present in CIFAR. For example, planes are often misclassified as 'bird' or 'ship' even without the attack because of the similar background of these images (i.e., sky is very similar to sea in many images). Indeed, the probability of predicting an airplane as a ship without the attack is 58%, and it only increases to 66% and 72% with FL-SIGN and FL-STD, respectively.

Almost the same remarks can be done on the other schemes FL-TOP and FL-CS. Indeed, with 10% of malicious nodes, the attack accuracy on the MNIST dataset for FL-SIGN is 77% and 61% less than FL-CS and FL-TOP, respectively. On the CIFAR dataset, the attack accuracy with 10% of malicious nodes for FL-SIGN is 8% and 6% less than FL-CS and FL-TOP, respectively.

As for in-backdoor attacks, FL-SIGN mitigates out-backdoor attacks because the adversary cannot scale up its update in order to increase its impact on the global model. In fact, we did not use boosting during Out-backdoor attacks and despite that FL-STD, FL-CS and FL-TOP have been more vulnerable. Therefore, even the use of the majority vote plays also an important role to be more robust.

		10%	20%	40%
FL-SIGN	Model accuracy	0.99	0.99	0.99
	Attack accuracy	0.19	0.87	0.99
FL-STD	Model accuracy	0.99	0.99	0.99
	Attack accuracy	0.92	0.99	0.99
FL-CS	Model accuracy	0.99	0.99	0.99
	Attack accuracy	0.83	0.95	0.99
FL-TOP	Model accuracy	0.99	0.99	0.99
	Attack accuracy	0.49	0.93	0.99

Tab. 6.7: Out-backdoors attack on FL-SIGN, FL-CS, FL-TOP and FL-STD with the MNIST dataset. η_{Adv} is set to 1 (no boosting). The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "0" to the targeted class "1" (attack accuracy). We choose the compression ratio r for FL-CS and FL-TOP such that each scheme reaches the same accuracy as FL-STD on the same settings but without the attack. Therefore, r is set to 5% and 0.5%, respectively.

		10%	20%	40%
FL-SIGN	Model accuracy	0.91	0.91	0.92
	Attack accuracy	0.66	0.74	0.93
FL-STD	Model accuracy	0.92	0.92	0.90
	Attack accuracy	0.72	0.86	0.95
FL-CS	Model accuracy	0.90	0.89	0.89
	Attack accuracy	0.72	0.80	0.96
FL-TOP	Model accuracy	0.91	0.92	0.91
	Attack accuracy	0.70	0.79	0.95

Tab. 6.8: Out-backdoors attack on FL-SIGN and FL-STD with the CIFAR dataset. η_{Adv} is set to 1 (no boosting). The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "airplane" to the targeted class "ship" (attack accuracy). We choose the compression ratio r for FL-CS and FL-TOP such that each scheme reaches the same accuracy as FL-STD on the same settings but without the attack. Therefore, r is set to 60% and 20%, respectively.

6.3 FL-SIGN-DP Security Analysis

Table 6.9, 6.10, 6.11 and 6.12 depict the accuracy of the in-backdoor class and the misclassification rate of the out-backdoor class (best values are chosen based on the global model accuracy over 200 rounds) when backdoor attacks are launched against FL-SIGN-DP schemes. Indeed, we investigate if FL-SIGN-DP is as robust as its non-private variant called FL-SIGN.

Malicious clients, whose fraction changes between 0.1 and 0.4, omit to add noise to their own updates at each round. Also, malicious clients collude by sharing their data and sending the same adversarial updates to the server. We use the configuration described in Table 8.2 except for γ which is decreased to 0.001. In Fashion-MNIST dataset, at all malicious nodes, all images of 'Sandal' are

relabelled to 'Sneaker' for In-backdoor, and all images of 'T-shirt/top' are relabelled to 'Trouser' for Out-backdoor attacks (only the malicious nodes have photos of 'T-shirt/top'). In addition, with FL-SIGN-DP, each malicious node calculates their updates, extracts the signs ($\mathsf{sign} : \mathbb{R}^n \rightarrow \{-1, 1\}^n$) and then uses a boosting parameter $\eta_{\mathsf{Adv}} = 5000$ to boost their updates before sending them back to the server for aggregation. Indeed, as all honest clients send the noisy update in FL-SIGN-DP, the noise together with encryption can conceal the manipulation of the malicious update vectors.

The results show that FL-SIGN-DP is less robust against backdoor attacks than FL-SIGN. On the MNIST dataset, model accuracy on the in-backdoor class is 0% for FL-SIGN-DP regardless of the number of malicious nodes, and larger than 97% and 95% for FL-SIGN, with 10% and 20% of malicious nodes, respectively. The same tendency holds for Fashion-MNIST. Out-backdoor attacks are especially effective on MNIST (see Table 6.11 and Table 6.12); here, the misclassification rate is more than 98% for FL-SIGN-DP and 0-99% for FL-SIGN. When we consider only 2% of malicious nodes with MNIST, the misclassification rate is 0% for FL-SIGN and 76% for FL-STD (without boosting) with a global model accuracy of 98% for both schemes. Indeed, FL-STD is vulnerable to the out-backdoor attack even if we have only a small number malicious node and without using any boosting. For MNIST and Fashion-MNIST, FL-SIGN is clearly superior to FL-SIGN-DP regarding all attacks.

Finally, random update attack and gradient ascent attack are mounted against FL-SIGN-DP. The same parameters are used as in the previous experiments. Malicious clients still collude and omit to add any noise to their own model updates. Instead, they boost their signs updates with FL-SIGN-DP ($\eta_{\mathsf{Adv}} = 5000$). The model fails to converge even if only 1% of all selected nodes are malicious at each round.

		10%	20%	40%
$\varepsilon = 1$	Model accuracy	0.89	0.90	0.90
	Accuracy on digit class '5'	0	0	0
$\varepsilon = 2$	Model accuracy	0.89	0.90	0.90
	Accuracy on digit class '5'	0	0	0
$\varepsilon = 4$	Model accuracy	0.89	0.90	0.90
	Accuracy on digit class '5'	0	0	0
FL-SIGN	Model accuracy	0.98	0.98	0.90
	Accuracy on digit class '5'	0.97	0.95	0

Tab. 6.9: In-backdoor attack on FL-SIGN-DP and FL-SIGN with MNIST dataset. The table depicts the global accuracy convergence and the accuracy of the label "5" which is under attack.

		10%	20%	40%
$\varepsilon = 1$	Model accuracy	0.77	0.79	0.80
	Accuracy on Sandal class	0	0	0
$\varepsilon = 2$	Model accuracy	0.77	0.79	0.80
	Accuracy on Sandal class	0	0	0
$\varepsilon = 4$	Model accuracy	0.77	0.79	0.80
	Accuracy on Sandal class	0	0	0
FL-SIGN	Model accuracy	0.83	0.84	0.79
	Accuracy on Sandal class	0.90	0.84	0

Tab. 6.10: In-backdoor attack on FL-SIGN and FL-SIGN-DP with Fashion-MNIST dataset. The table depicts the global accuracy convergence and the accuracy of the label "Sandal" which is under attack.

6.4 Conclusion

FL-SIGN is almost as accurate as FL-STD but incurs less communication overhead and has better resiliency against both security and privacy attacks (see Section 4.1.3 and 6.2). FL-TOP and FL-CS are also less robust against poisoning attacks compared to FL-SIGN. In fact, FL-SIGN is more robust due to: (1) the quantization which mitigates the adversarial updates and (2) the majority vote which implies to take a more robust decision regarding how to update of each weight.

		2%	10%	20%	40%
$\varepsilon = 1$	Model accuracy	0.98	0.98	0.99	0.99
	Attack accuracy	0.98	0.99	0.99	1
$\varepsilon = 2$	Model accuracy	0.98	0.98	0.98	0.99
	Attack accuracy	0.99	0.98	0.99	0.99
$\varepsilon = 4$	Model accuracy	0.98	0.98	0.99	0.99
	Attack accuracy	0.99	0.99	0.99	0.99
FL-SIGN	Model accuracy	0.98	0.98	0.98	0.99
	Attack accuracy	0	0.97	0.99	0.99

Tab. 6.11: Out-backdoors attack on FL-SIGN-DP and FL-SIGN with MNIST dataset. The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "0" to the targeted class "1" (attack accuracy).

		10%	20%	40%
$\varepsilon = 1$	Model accuracy	0.87	0.88	0.90
	Attack accuracy	0.78	0.81	0.87
$\varepsilon = 2$	Model accuracy	0.88	0.88	0.90
	Attack accuracy	0.78	0.82	0.85
$\varepsilon = 4$	Model accuracy	0.87	0.89	0.90
	Attack accuracy	0.78	0.81	0.86
FL-SIGN	Model accuracy	0.87	0.88	0.90
	Attack accuracy	0	0.12	0.83

Tab. 6.12: Out-backdoors attack on FL-SIGN-DP and FL-SIGN with Fashion-MNIST dataset. The table displays the global model accuracy as well as the model's prediction rate to misclassify the out-backdoor class "T-shirt/Top" to the targeted class "Trouser" (attack accuracy).

Notice that, the "Curse of Dimensionality" assumption defined in [CSSH19] which claims that large models, with high dimensional parameter vectors are more vulnerable to security attacks, is not always true. Indeed, we have shown in our results that the bandwidth efficient FL-TOP and FL-CS schemes are sometimes less robust than the non-compressed scheme FL-STD (see Table 6.5).

Seemingly, there is a possible trade-off between differential privacy and robustness against security attacks. One possible explanation is that differential privacy requires to randomize every value of the update vector so much that their aggregates become easier to manipulate. As malicious clients omit to add any noise to their own model updates, the attacked DP protocols essentially turn into Random Update Attacks, where honest clients send almost uniformly random signs and malicious clients transfer non-noisy, boosted updates to the aggregator. As FL-SIGN converges with Random Update Attack even with limited number of honest nodes, the malicious nodes in FL-SIGN-DP can also degrade model performance or inject backdoors for the very same reason. The smaller ε is the more uniform every coordinate's distribution will be, and the larger impact a malicious client has on the aggregate.

7 Conclusion and perspectives

7.1 Concluding Remarks

As described in this thesis Federated Learning is a very promising approach but suffers from a high bandwidth cost. Furthermore, although it improves privacy, it does not provide any theoretical privacy-preserving guarantee. This thesis reports some of the works of my Ph.D. studies where we tried to address some of these issues.

The first part of this thesis proposes three bandwidth efficient Federated Learning schemes. The first solution is called FL-SIGN and it is a quantization approach. FL-SIGN reduces the number of bits per parameter by a factor of 32. Therefore, instead of sending 32 bits per parameter, we only have to send 1 bit: +1/-1 to either increase or decrease the weight. The second solution FL-CS leverages the sparsity of the updates which are sent from the client to the server. By using Compressive sensing theory [JV10], which allows to reconstruct a sparse data (i.e., image or signal) from some sampled points in it, we have shown that it is possible to learn a model as accurate as the model learned via standard federated learning. Moreover, FL-CS allows to compress each update up to 95% compared to the standard scheme. FL-TOP for its part harnesses the ability of the model to adapt its weights under constraint. By updating only the weights which are more likely to be updated during the training process, we have shown that it is possible to compress the model until 99.9%. Moreover, our previous proposals FL-SIGN and FL-CS are bandwidth efficient only during upstream traffic (from clients to server), however, FL-TOP is bandwidth efficient in both upstream and downstream directions due to the fact that the same proportion of weights are updated during all the learning process while keeping all the remaining ones constant. Therefore, this protocol seems to be well adapted to an environment with energy-constrained devices as the batteries of such devices are highly sensitive to the network communications.

The second part of this thesis defines Differentially Private versions of FL-SIGN, FL-CS and FL-TOP. The private extension of FL-SIGN called FL-SIGN-DP provides client-level Differential Privacy (DP) and aims at protecting any information that is unique to a client's training data. We show that our DP learning protocol produces models with an accuracy comparable to its non-private version even with stringent privacy guarantees (eg., $\epsilon = 1$). The private extension of the federated learning scheme called FL-STD has performance slightly better than FL-SIGN-DP (this difference tends to be reduced as we increase the privacy protection (reducing ϵ).), however the latter has 30-40% less communication cost than FL-STD-DP and FL-STD. The bandwidth efficiency of our DP algorithm is due to the use of a novel discretized and distributed version of the Gaussian Mechanism. Therefore, the noise values are discrete and can be encoded with fewer bits than if they came from a continuous Gaussian distribution.

The utility-privacy trade-off is one of the challenge that we face when we use DP. Having a good privacy guarantee (small ϵ) is often synonymous with high accuracy decrease and vice-versa. To have a differentially private solution, we have to noise the updates. The required noise is generally sampled from a Gaussian distribution with mean 0 and variance $S^2\sigma^2$. S is the sensitivity of the model and it is generally proportional to its size. By reducing the size of the model we also reduce its sensitivity and therefore the required noise for a differentially private guarantee. This is exactly what FL-CS-DP and FL-TOP-DP leverage in order to improve the utility of the model. Both of them outperform the FL-STD-DP scheme in both accuracy and bandwidth efficiency. Moreover, in FL-CS-DP the noise is added on the first discrete cosine transform frequencies of each update which are generally larger than the other frequencies and only few weights are updated in FL-TOP-DP during the learning process

which make their values larger compared to the case when they are trained under the standard learning scheme FL-STD-DP. Therefore, FL-CS-DP and FL-TOP-DP increase the value-to-noise to improve the model's accuracy.

Finally, Chapter 6 evaluates the robustness of the proposed schemes in an adversarial environment. First, we have evaluated the robustness of our non private schemes: FL-SIGN, FL-CS and FL-TOP against poisoning attacks.The poisoning attacks can be either targeted where the adversary aims at reducing the accuracy of the model regarding a specific class, or untargeted and in that case the adversary aims at reducing the global model accuracy without distinction between classes. Our results show that FL-SIGN has better resiliency against both targeted and untargeted attacks compared to FL-CS, FL-TOP and FL-STD even by considering strong adversaries which collude by sharing their data and sending the same updates. FL-SIGN mitigates out-backdoor attacks because the adversary cannot scale up its update in order to increase its impact on the global model. Also, the majority vote allows to take a better decision on how each weight should be updated. This straightforward approach can be also used with secure aggregation which is not the case of most techniques designed to detect the malicious updates or to choose the honest updates. Indeed, they generally require access to each individual participant's update which is therefore incompatible with secure aggregation. After that, we evaluated the robustness of the private extension of FL-SIGN in order to see if it is as robust. However, the results have shown that FL-SIGN-DP is not robust against poisoning attacks due to the added noise requires by the differential privacy scheme. Our result introduces a possible trade-off between differential privacy and robustness against security attacks.

7.2 Future Research Directions

- **Designing Differentially Private and Secure Federated Learning Scheme.** This research indicates that there might exist a robustness/privacy tradeoff and we would like to explore this further. Using Game Theory algorithms might be an interesting direction to investigate. Such algorithms aim at increasing a profit or minimizing a cost. If we consider that our profit is to increase the utility of the model, such algorithms may help the server to select only or at least more often honest participants than malicious ones. For example, we can model the selection of an honest set of clients as a multi-armed bandit problem [ACF02; BF85]. A Multiarmed bandit problem models a casino with a slot machine of k arms, where pulling each arm results in a different and unknown expected payoff. The goal is to sequentially select the optimal sequence of slot machine arms to pull to maximize the expected total reward. Considering each selected set of client $\mathbb{K}$ at each round as an arm, that may or may not provide a reward (improve model accuracy for example) when it is used.

- **Improving the Utility-Privacy Tradeoff.** Using differential privacy has a cost and may highly reduce the utility (accuracy) of the model. To reduce this negative effect of DP on the accuracy, we reduced the sensitivity of the model by reducing its size and we also increased the value-to-noise. However, there is also other possibilities to reduce the noise either by: 1) reducing the number of federated' rounds or 2) reducing the sampling probability of any participant. In Federated Learning, each selected client at a specific round, updates the received model based on his data only during few, limited SGD iterations. Indeed, if the selected clients decide to run an unlimited number of SGD iterations[1], the aggregated model tends to diverge, due to the fact that each client's update will be highly personalized on its dataset. Therefore, it is interesting to find how we can run the maximum number of SGD iterations at the client side without impacting the convergence of the model after the aggregation. This might speed up the training process and therefore decrease the noise. One possible solution to reduce the sampling probability is to leverage the large amount of public data freely available online. More specifically, we can add some simulated clients with public data in order to reduce the sampling probability and therefore to reduce the noise. However, we should find a way to either make the updates of the simulated clients valuable even if they use public data or limit their impact on the learning

[1] but of course without overfitting on their local testing data

process otherwise. Note that this approach remains valid even for the centralized learning scenario with a record-level privacy guarantee.

- **Private and Secure Federated Learning Based on New Approaches.** Most of the research on federated learning focus on the standard federated learning approach called Horizontally Federated Learning [KC04; SS15; MMRHA16], where the same features are shared between participants, but they differ on their records. However, the security and privacy aspects of the Vertical Federated Learning (VFL) and Federated Transfer Learning (FTL)[2] were not studied yet. Therefore, it is interesting to evaluate the different poisoning attacks and inference attacks on the other approaches. Moreover, proper attacks and solutions might be designed for the non-studied approaches. Similarly, It is interesting to study the privacy and robustness of the recent AI approaches like Self-supervised learning and Graph neural networks in federated learning context. Indeed, Self-supervised is a new approach and sub-class of unsupervised learning. According to Professor Yann LeCun and Professor Yoshua Bengio, it will be the future of AI as it may allow machines to develop "common sense" just like humans [21b; 21a]. Graph Neural Networks are also becoming increasingly popular as these allow better explainability of the model's prediction. It is therefore important to study their privacy and security properties in a collaborative learning environment as with Federated Learning.

- **Detection of malicious entities.** It is interesting to provide some mechanisms to either detect malicious servers or participants, which aims at inferring sensitive information or at degrading the accuracy of the model via poisoning attacks. However, this solution should take into account that a strong adversary can adapt and behave malicious only during some rounds. Therefore, by switching from honest to malicious behaviour, it is really challenging to detect such adversaries.

- **Evaluating the contribution of each participant under DP.** Different solutions allow to evaluate the contribution of malicious entities under federated learning settings [Pej20; FVL21] to prevent Free-rider Attacks [FVL21]. Indeed, in such attacks, the adversary's goal is to obtain the final aggregated model without actually contributing with any data. However, previous solutions do not consider DP. Therefore, it is interesting to find a way to do it when DP is used. It is particularly challenging because by definition using DP for a client-level differential privacy does not allow distinguishing between clients.

- **Evaluating the impact of compression on the model's performance.** It is interesting to study the negative impact of the compression on the model's performance. This should include more aspects than only considering the potential loss in average accuracy. In [HCCDF20], the authors show that compression disproportionately impacts model performance on the underrepresented long-tail of the data distribution. Therefore, it seems that compression has a negative impact on fairness. We should then inverstigate all the possible aspects which may be impacted negatively by compression.

[2] refer to Section 2.1 for details.

Bibliography

[20] *Microsoft SEAL (release 3.6)*. https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA. Nov. 2020 (cit. on p. 17).

[21a] *Self-supervised learning: The plan to make deep learning data-efficient*. https://bdtechtalks.com/2020/03/23/yann-lecun-self-supervised-learning/. 3/05/2021 (cit. on p. 63).

[21b] *Yann LeCun and Yoshua Bengio: Self-supervised learning is the key to human-level intelligence*. https://venturebeat.com/2020/05/02/yann-lecun-and-yoshua-bengio-self-supervised-learning-is-the-key-to-human-level-intelligence/. 3/05/2021 (cit. on p. 63).

[Aba+16] Martin Abadi, Andy Chu, Ian Goodfellow, et al. „Deep Learning with Differential Privacy". In: *ACM CCS*. Vienna, Austria: ACM, 2016, pp. 308–318 (cit. on pp. 10, 16, 38, 41, 46, 75, 78).

[ÁC11] Gergely Ács and Claude Castelluccia. „I Have a DREAM! (DiffeRentially privatE smArt Metering)". In: *Information Hiding - 13th International Conference, IH 2011*. 2011, pp. 118–132 (cit. on pp. 35–37, 41, 46).

[ACF02] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. „Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* 47.2 (2002), pp. 235–256 (cit. on p. 62).

[AG07] Galen Andrew and Jianfeng Gao. „Scalable training of L 1-regularized log-linear models". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 33–40 (cit. on pp. 12, 26).

[AG19] M. M. Amiri and D. Gündüz. „Machine Learning at the Wireless Edge: Distributed Stochastic Gradient Descent Over-the-Air". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. 2019, pp. 1432–1436 (cit. on p. 17).

[AG20] M. M. Amiri and D. Gündüz. „Federated Learning Over Wireless Fading Channels". In: *IEEE Transactions on Wireless Communications* 19.5 (2020), pp. 3546–3557 (cit. on p. 17).

[AGLTV17] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. „QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, et al. Vol. 30. Curran Associates, Inc., 2017 (cit. on p. 17).

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. „The multiplicative weights update method: a meta-algorithm and applications". In: *Theory of Computing* 8.1 (2012), pp. 121–164 (cit. on p. 20).

[Ahm91] Nasir Ahmed. „How I came up with the discrete cosine transform". In: *Digit. Signal Process.* 1.1 (1991), pp. 4–5 (cit. on pp. 26, 27).

[Ako17] Josephine Akosa. „Predictive accuracy: a misleading performance measure for highly imbalanced data". In: *Proceedings of the SAS Global Forum*. 2017, pp. 2–5 (cit. on p. 81).

[ANR74] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. „Discrete cosine transform". In: *IEEE transactions on Computers* 100.1 (1974), pp. 90–93 (cit. on pp. 26, 27).

[APR19] Paras Malik Amisha, Monika Pathania, and Vyas Kumar Rathaur. „Overview of artificial intelligence in medicine". In: *Journal of family medicine and primary care* 8.7 (2019), p. 2328 (cit. on p. 1).

[Ava+18] Anand Avati, Kenneth Jung, Stephanie Harman, et al. „Improving palliative care with deep learning". In: *BMC Medical Informatics and Decision Making* 18.4 (Dec. 2018), p. 122 (cit. on pp. 79, 80).

[BBG19] Gilad Baruch, Moran Baruch, and Yoav Goldberg. „A Little Is Enough: Circumventing Defenses For Distributed Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, et al. Vol. 32. Curran Associates, Inc., 2019 (cit. on p. 20).

[BCMC19] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. „Analyzing federated learning through an adversarial lens". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 634–643 (cit. on pp. 18–20, 52, 53).

[BDA13] Mohamed Bekkar, Hassiba Djema, and T.A. Alitouche. „Evaluation measures for models assessment over imbalanced data sets“. In: *Journal of Information Engineering and Applications* 3 (Jan. 2013), pp. 27–38 (cit. on pp. 28, 81).

[BEGS17] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. „Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent“. In: *NIPS*. 2017, pp. 119–129 (cit. on pp. 2, 18–20, 52).

[BF85] Donald A Berry and Bert Fristedt. „Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)“. In: *London: Chapman and Hall* 5.71-87 (1985), pp. 7–7 (cit. on p. 62).

[BLPL07] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. „Greedy layer-wise training of deep networks“. In: *Advances in neural information processing systems*. 2007, pp. 153–160 (cit. on p. 17).

[BNL12] Battista Biggio, Blaine Nelson, and Pavel Laskov. „Poisoning Attacks against Support Vector Machines“. In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML'12. Edinburgh, Scotland: Omnipress, 2012, pp. 1467–1474 (cit. on pp. 2, 18).

[Bon+16] K. A. Bonawitz, Vladimir Ivanov, Ben Kreuter, et al. „Practical Secure Aggregation for Federated Learning on User-Held Data“. In: *NIPS Workshop on Private Multi-Party Machine Learning*. 2016 (cit. on pp. 4, 36, 37, 41, 46).

[BOSB10] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. „The balanced accuracy and its posterior distribution“. In: *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, pp. 3121–3124 (cit. on pp. 28, 81).

[BVHES18] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. „How To Backdoor Federated Learning“. In: *CoRR* abs/1807.00459 (2018). arXiv: `1807.00459` (cit. on pp. 18, 19).

[BWAA18] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. „signSGD: Compressed Optimisation for Non-Convex Problems“. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, Oct. 2018, pp. 560–569 (cit. on pp. 16, 17, 20, 24).

[BZAA18] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. „signSGD with Majority Vote is Communication Efficient And Byzantine Fault Tolerant“. In: *CoRR* abs/1810.05291 (2018). arXiv: `1810.05291` (cit. on pp. 2, 18–20, 24, 52–54, 76–78).

[Can08] Emmanuel Candès. „The restricted isometry property and its implications for compressed sensing“. In: *Compte Rendus de l'Academie des Sciences* 346 (May 2008), pp. 589–592 (cit. on p. 12).

[Cao20] Longbing Cao. „AI in Finance: A Review“. In: *Available at SSRN 3647625* (2020) (cit. on p. 1).

[CCF02] Moses Charikar, Kevin Chen, and Martin Farach-Colton. „Finding frequent items in data streams“. In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2002, pp. 693–703 (cit. on p. 18).

[CDS01] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. „Atomic decomposition by basis pursuit“. In: *SIAM review* 43.1 (2001), pp. 129–159 (cit. on pp. 11, 41).

[Cho+15a] François Chollet et al. *Keras*. `https://keras.io`. 2015 (cit. on p. 82).

[Cho+15b] François Chollet et al. *Keras datasets*. `https://keras.io/datasets/`. 2015 (cit. on p. 79).

[Cho+15c] François Chollet et al. *Keras Early Stopping*. `https://keras.io/api/callbacks/early_stopping/`. 2015 (cit. on p. 4).

[Cho+15d] François Chollet et al. *Keras optimizers*. `https://keras.io/optimizers/`. 2015 (cit. on p. 25).

[Cho+19] Olivia Choudhury, Aris Gkoulalas-Divanis, Theodoros Salonidis, et al. *Differential Privacy-enabled Federated Learning for Sensitive Health Data*. 2019. arXiv: `1910.02578 [cs.LG]` (cit. on p. 2).

[CHSEB16] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. *Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1*. 2016. arXiv: `1602.02830 [cs.LG]` (cit. on p. 18).

[CKS20] Clément L Canonne, Gautam Kamath, and Thomas Steinke. „The Discrete Gaussian for Differential Privacy“. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 15676–15688 (cit. on pp. 16, 38).

[CLLLS17] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. „Targeted backdoor attacks on deep learning systems using data poisoning“. In: *arXiv preprint arXiv:1712.05526* (2017) (cit. on p. 18).

[CM09] Kamalika Chaudhuri and Claire Monteleoni. „Privacy-preserving logistic regression“. In: *Advances in neural information processing systems*. 2009, pp. 289–296 (cit. on p. 16).

[CMS11] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. „Differentially private empirical risk minimization“. In: *Journal of Machine Learning Research* 12.Mar (2011), pp. 1069–1109 (cit. on p. 16).

[CRT06] Emmanuel J Candès, Justin Romberg, and Terence Tao. „Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information“. In: *IEEE Transactions on information theory* 52.2 (2006), pp. 489–509 (cit. on pp. 3, 11, 17).

[CSSH19] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. „Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer“. In: *arXiv preprint arXiv:1912.11279* (2019) (cit. on pp. 4, 20, 59).

[CT05] Emmanuel J Candes and Terence Tao. „Decoding by linear programming“. In: *IEEE transactions on information theory* 51.12 (2005), pp. 4203–4215 (cit. on p. 12).

[CT06] E. J. Candes and T. Tao. „Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?“ In: *IEEE Transactions on Information Theory* 52.12 (2006), pp. 5406–5425 (cit. on pp. 3, 11, 17, 26).

[CUA19] Marta TERRON CUADRADO. *ICD-9-CM: International Classification of Diseases, Ninth Revision, Clinical Modification*. `https://ec.europa.eu/cefdigital/wiki/display/EHSEMANTIC/ICD-9-CM%3A+International+Classification+of+Diseases%2C+Ninth+Revision%2C+Clinical+Modification`. 2019 (cit. on p. 83).

[CW17] Nicholas Carlini and David Wagner. „Towards evaluating the robustness of neural networks“. In: *2017 ieee symposium on security and privacy (sp)*. IEEE. 2017, pp. 39–57 (cit. on p. 19).

[Dia+17] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, et al. „Being Robust (in High Dimensions) Can Be Practical“. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 999–1008 (cit. on p. 20).

[DJW13] John C Duchi, Michael I Jordan, and Martin J Wainwright. „Local privacy and statistical minimax rates“. In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE. 2013, pp. 429–438 (cit. on p. 16).

[Don06] David L Donoho. „Compressed sensing“. In: *IEEE Transactions on information theory* 52.4 (2006), pp. 1289–1306 (cit. on pp. 3, 11, 17).

[DR14] Cynthia Dwork and Aaron Roth. „The Algorithmic Foundations of Differential Privacy“. In: *Foundations and Trends in Theoretical Computer Science* 9.3–4 (2014) (cit. on pp. 2, 3, 10, 11, 35).

[EGR18] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. „The Hidden Vulnerability of Distributed Learning in Byzantium“. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 3521–3530 (cit. on p. 20).

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. „RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response“. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM, 2014, pp. 1054–1067 (cit. on pp. 4, 36).

[FC18] Jonathan Frankle and Michael Carbin. „The Lottery Ticket Hypothesis: Training Pruned Neural Networks“. In: (2018) (cit. on p. 18).

[FGLB18] A. Fejza, P. Genevès, N. Layaïda, and J. Bosson. „Scalable and Interpretable Predictive Models for Electronic Health Records“. In: *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. Oct. 2018, pp. 341–350 (cit. on pp. 79, 80).

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. „Model inversion attacks that exploit confidence information and basic countermeasures“. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 1322–1333 (cit. on p. 15).

[FS97] Yoav Freund and Robert E Schapire. „A decision-theoretic generalization of on-line learning and an application to boosting“. In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139 (cit. on p. 20).

[FVL21] Yann Fraboni, Richard Vidal, and Marco Lorenzi. *Free-rider Attacks on Model Aggregation in Federated Learning*. 2021. arXiv: `2006.11901 [cs.LG]` (cit. on p. 63).

[FYB18] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. „Mitigating sybils in federated learning poisoning“. In: *arXiv preprint arXiv:1808.04866* (2018) (cit. on pp. 18, 20).

[GBDM20] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. *Inverting Gradients – How easy is it to break privacy in federated learning?* 2020. arXiv: 2003.14053 [cs.CV] (cit. on pp. 2, 15, 35).

[GDG17] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. „Badnets: Identifying vulnerabilities in the machine learning model supply chain". In: *arXiv preprint arXiv:1708.06733* (2017) (cit. on p. 18).

[GK07] Naveen Garg and Jochen Könemann. „Faster and simpler algorithms for multicommodity flow and other fractional packing problems". In: *SIAM Journal on Computing* 37.2 (2007), pp. 630–652 (cit. on p. 20).

[GKN17] Robin C. Geyer, Tassilo Klein, and Moin Nabi. „Differentially Private Federated Learning: A Client Level Perspective". In: *CoRR* abs/1712.07557 (2017). arXiv: 1712.07557 (cit. on pp. 16, 81).

[Goo+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. „Generative Adversarial Nets". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 2672–2680 (cit. on p. 15).

[Han+16] Song Han, Jeff Pool, Sharan Narang, et al. *DSD: Dense-Sparse-Dense Training for Deep Neural Networks*. 2016. arXiv: 1607.04381 [cs.CV] (cit. on pp. 18, 29, 30).

[HAP17] Briland Hitaj, Giuseppe Ateniese, and Fernando 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*. 2017, pp. 603–618 (cit. on p. 15).

[HCB16] Jihun Hamm, Paul Cao, and Mikhail Belkin. „Learning Privately from Multiparty Data". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 555–563 (cit. on p. 16).

[HCCDF20] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome. *What Do Compressed Deep Neural Networks Forget?* 2020. arXiv: 1911.05248 [cs.LG] (cit. on p. 63).

[HG09] Haibo He and Edwardo A Garcia. „Learning from imbalanced data". In: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284 (cit. on p. 80).

[HJNRT11] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. „Adversarial machine learning". In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. 2011, pp. 43–58 (cit. on p. 19).

[HRBL07] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, Oct. 2007 (cit. on p. 79).

[IRUSA+19] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, Raman Arora, et al. „Communication-efficient distributed sgd with sketching". In: *Advances in Neural Information Processing Systems*. 2019, pp. 13144–13154 (cit. on pp. 18, 27).

[Iye+19] Roger Iyengar, Joseph P Near, Dawn Song, et al. „Towards practical differentially private convex optimization". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 299–316 (cit. on p. 16).

[Jag+18] Matthew Jagielski, Alina Oprea, Battista Biggio, et al. „Manipulating machine learning: Poisoning attacks and countermeasures for regression learning". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 19–35 (cit. on pp. 18, 19).

[JALP21] Y. -S. Jeon, M. M. Amiri, J. Li, and H. V. Poor. „A Compressive Sensing Approach for Federated Learning Over Massive MIMO Communication Systems". In: *IEEE Transactions on Wireless Communications* 20.3 (2021), pp. 1990–2004 (cit. on p. 17).

[JE19] Bargav Jayaraman and David Evans. „Evaluating differentially private machine learning in practice". In: *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 2019, pp. 1895–1912 (cit. on p. 16).

[JHHDW20] Richeng Jin, Yufan Huang, Xiaofan He, Huaiyu Dai, and Tianfu Wu. „Stochastic-Sign SGD for Federated Learning with Theoretical Guarantees". In: *CoRR* abs/2002.10940 (2020). arXiv: 2002.10940 (cit. on p. 16).

[JKT12] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. „Differentially private online learning". In: *Conference on Learning Theory*. JMLR Workshop and Conference Proceedings. 2012, pp. 24–1 (cit. on p. 16).

[JT14] Prateek Jain and Abhradeep Guha Thakurta. „(Near) dimension independent risk bounds for differentially private learning". In: *International Conference on Machine Learning*. PMLR. 2014, pp. 476–484 (cit. on p. 16).

[JV10] Laurent Jacques and Pierre Vandergheynst. *Compressed sensing: When sparsity meets sampling*. Tech. rep. Wiley-Blackwell, 2010 (cit. on pp. 11, 12, 17, 41, 61).

[Kai+19] Peter Kairouz, H Brendan McMahan, Brendan Avent, et al. „Advances and open problems in federated learning“. In: *arXiv preprint arXiv:1912.04977* (2019) (cit. on pp. 17, 18).

[KB14] Diederik P. Kingma and Jimmy Ba. „Adam: A Method for Stochastic Optimization“. In: *CoRR* abs/1412.6980 (2014) (cit. on p. 25).

[KC04] Murat Kantarcioglu and Chris Clifton. „Privacy-preserving distributed mining of association rules on horizontally partitioned data“. In: *IEEE transactions on knowledge and data engineering* 16.9 (2004), pp. 1026–1037 (cit. on pp. 10, 63).

[KL17] Pang Wei Koh and Percy Liang. „Understanding black-box predictions via influence functions“. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1885–1894 (cit. on p. 18).

[KNH] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. In: () (cit. on p. 79).

[Kon+16] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, et al. „Federated Learning: Strategies for Improving Communication Efficiency“. In: *CoRR* abs/1610.05492 (2016). arXiv: 1610.05492 (cit. on pp. 17, 24).

[KRSJ19] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. „Error Feedback Fixes SignSGD and other Gradient Compression Schemes“. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 3252–3261 (cit. on pp. 12, 13).

[LBLL09] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. „Exploring strategies for training deep neural networks.“ In: *Journal of machine learning research* 10.1 (2009) (cit. on p. 17).

[LC10] Yann LeCun and Corinna Cortes. „MNIST handwritten digit database“. In: (2010) (cit. on p. 79).

[LCYC20] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. „FedSel: Federated SGD Under Local Differential Privacy with Top-k Dimension Selection“. In: *Lecture Notes in Computer Science* (2020) (cit. on p. 16).

[LHMWD18] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. „Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training“. In: *International Conference on Learning Representations, ICLR 2018*. 2018 (cit. on p. 17).

[Li+14] Qi Li, Yaliang Li, Jing Gao, et al. „Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation“. In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 2014, pp. 1187–1198 (cit. on p. 20).

[Li+20] Ang Li, Jingwei Sun, Binghui Wang, et al. *LotteryFL: Personalized and Communication-Efficient Federated Learning with Lottery Ticket Hypothesis on Non-IID Datasets*. 2020. arXiv: 2008.03371 [cs.LG] (cit. on p. 18).

[Liu+18] Yingqi Liu, Shiqing Ma, Yousra Aafer, et al. „Trojaning Attack on Neural Networks“. In: *NDSS*. 2018 (cit. on p. 18).

[Liu20] Simon Y. Liu. „Artificial Intelligence (AI) in Agriculture“. In: *IT Professional* 22.3 (2020), pp. 14–15 (cit. on p. 1).

[LW19] Daliang Li and Junpu Wang. „Fedmd: Heterogenous federated learning via model distillation“. In: *arXiv preprint arXiv:1910.03581* (2019) (cit. on p. 10).

[Lyu+19] Lingjuan Lyu, Jiangshan Yu, Karthik Nandakumar, et al. „Towards fair and decentralized privacy-preserving deep learning with blockchain“. In: *arXiv preprint arXiv:1906.01167* (2019), pp. 1–13 (cit. on p. 10).

[LYY20] Lingjuan Lyu, Han Yu, and Qiang Yang. „Threats to federated learning: A survey“. In: *arXiv preprint arXiv:2003.02133* (2020) (cit. on pp. 10, 19).

[LZWY11] Yang D. Li, Zhenjie Zhang, Marianne Winslett, and Yin Yang. „Compressive Mechanism: Utilizing Sparse Representation in Differential Privacy“. In: *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*. WPES '11. Chicago, Illinois, USA: Association for Computing Machinery, 2011, pp. 177–182 (cit. on p. 17).

[Mcd+12] Margaret Mcdonald, Timothy Peng, Sridevi Sridharan, et al. „Automating the medication regimen complexity index“. In: *Journal of the American Medical Informatics Association : JAMIA* 20 (Dec. 2012) (cit. on p. 80).

[Mir12] Ilya Mironov. „On Significance of the Least Significant Bits for Differential Privacy“. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. CCS '12. Raleigh, North Carolina, USA: Association for Computing Machinery, 2012, pp. 650–661 (cit. on p. 16).

[MKAV11] Hossein Mamaghanian, Nadia Khaled, David Atienza, and Pierre Vandergheynst. „Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes“. In: *IEEE Transactions on Biomedical Engineering* 58.9 (2011), pp. 2456–2466 (cit. on p. 17).

[MMB16] Christopher A Metzler, Arian Maleki, and Richard G Baraniuk. „From denoising to compressed sensing“. In: *IEEE Transactions on Information Theory* 62.9 (2016), pp. 5117–5144 (cit. on p. 17).

[MMRHA16] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. „Communication-Efficient Learning of Deep Networks from Decentralized Data“. In: *AISTATS*. 2016 (cit. on pp. 1, 9, 10, 63, 78).

[Mo+21] Fan Mo, Hamed Haddadi, Kleomenis Katevas, et al. „PPFL: Privacy-Preserving Federated Learning with Trusted Execution Environments“. In: *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys '21. Virtual Event, Wisconsin: Association for Computing Machinery, 2021, pp. 94–108 (cit. on p. 17).

[Mor16] Ajinkya More. „Survey of resampling techniques for improving classification performance in unbalanced datasets“. In: *arXiv preprint arXiv:1608.06048* (2016) (cit. on p. 80).

[MR14] Rupa Makadia and Patrick B. Ryan. „Transforming the Premier Perspective® Hospital Database into the Observational Medical Outcomes Partnership (OMOP) Common Data Model“. In: *EGEMS*. 2014 (cit. on p. 79).

[MRTZ18] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. „Learning Differentially Private Recurrent Language Models“. In: *International Conference on Learning Representations*. 2018 (cit. on pp. 16, 81).

[MSDS19] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. „Exploiting unintended feature leakage in collaborative learning“. In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 691–706 (cit. on pp. 2, 15, 35, 36, 78).

[MTZ19] Ilya Mironov, Kunal Talwar, and Li Zhang. „Rényi Differential Privacy of the Sampled Gaussian Mechanism“. In: *CoRR* abs/1908.10530 (2019). arXiv: 1908.10530 (cit. on pp. 38, 41, 46).

[Muñ+17] Luis Muñoz-González, Battista Biggio, Ambra Demontis, et al. „Towards poisoning of deep learning algorithms with back-gradient optimization“. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. 2017, pp. 27–38 (cit. on pp. 18, 19).

[MW17] Daniele Micciancio and Michael Walter. „Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time“. In: *Advances in Cryptology - CRYPTO 2017*. 2017, pp. 455–485 (cit. on pp. 37, 38, 75, 76).

[MZ15] Shike Mei and Xiaojin Zhu. „Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners“. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 2871–2877 (cit. on pp. 2, 18).

[Nar18] Sarang Narkhede. *Understanding AUC - ROC Curve*. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. 2018 (cit. on pp. 27, 43).

[Nat95] Balas Kausik Natarajan. „Sparse approximate solutions to linear systems“. In: *SIAM journal on computing* 24.2 (1995), pp. 227–234 (cit. on p. 11).

[NHC21] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. *Toward Robustness and Privacy in Federated Learning: Experimenting with Local and Central Differential Privacy*. 2021. arXiv: 2009.03561 [cs.CR] (cit. on p. 17).

[NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. „Can Homomorphic Encryption Be Practical?“ In: *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*. CCSW '11. Chicago, Illinois, USA: Association for Computing Machinery, 2011, pp. 113–124 (cit. on p. 17).

[NSH18] Milad Nasr, Reza Shokri, and Amir Houmansadr. „Machine learning with membership privacy using adversarial regularization“. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 634–646 (cit. on p. 17).

[NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. „Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning“. In: *IEEE Symposium on Security and Privacy, 2019*. 2019, pp. 739–753 (cit. on pp. 2, 15, 35, 52).

[Oli06] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006 (cit. on p. 82).

[PAEGT16] Nicolas Papernot, Martı'n Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. „Semi-supervised knowledge transfer for deep learning from private training data“. In: *arXiv preprint arXiv:1610.05755* (2016) (cit. on p. 16).

[Pej20] Bal'azs Pej'o. „The Good, The Bad, and The Ugly: Quality Inference in Federated Learning“. In: *ArXiv* abs/2007.06236 (2020) (cit. on p. 63).

[PRR10] Manas A Pathak, Shantanu Rane, and Bhiksha Raj. „Multiparty Differential Privacy via Aggregation of Locally Trained Classifiers.“ In: *NIPS*. Citeseer. 2010, pp. 1876–1884 (cit. on p. 16).

[PST95] Serge A Plotkin, David B Shmoys, and Éva Tardos. „Fast approximation algorithms for fractional packing and covering problems“. In: *Mathematics of Operations Research* 20.2 (1995), pp. 257–301 (cit. on p. 20).

[PWD17] NhatHai Phan, Xintao Wu, and Dejing Dou. „Preserving differential privacy in convolutional deep belief networks“. In: *Machine learning* 106.9 (2017), pp. 1681–1704 (cit. on p. 16).

[PWWD16] NhatHai Phan, Yue Wang, Xintao Wu, and Dejing Dou. „Differential Privacy Preservation for Deep Auto-Encoders: An Application of Human Behavior Prediction“. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 1309–1316 (cit. on p. 16).

[Ra18] Alvin Rajkomar and al. „Scalable and accurate deep learning with electronic health records“. In: *npj Digital Medicine* 1.1 (2018). url, An earlier version appeared in eprint arXiv:1801.07860, p. 18 (cit. on p. 80).

[Rot+20] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, et al. „FetchSGD: Communication-Efficient Federated Learning with Sketching“. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 8253–8265 (cit. on p. 18).

[Rub+09] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, et al. „Stealthy poisoning attacks on PCA-based anomaly detectors“. In: *ACM SIGMETRICS Performance Evaluation Review* 37.2 (2009), pp. 73–74 (cit. on pp. 2, 18).

[RY14] K Ramamohan Rao and Ping Yip. *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014 (cit. on p. 26).

[SCS13] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. „Stochastic gradient descent with differentially private updates“. In: *2013 IEEE Global Conference on Signal and Information Processing*. IEEE. 2013, pp. 245–248 (cit. on p. 16).

[SDBR15] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. „Striving for Simplicity: The All Convolutional Net“. In: *ICLR (workshop track)*. 2015 (cit. on p. 78).

[SFDLY14] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. „1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs“. In: *INTERSPEECH 2014*. 2014, pp. 1058–1062 (cit. on p. 17).

[Sha+18] Ali Shafahi, W Ronny Huang, Mahyar Najibi, et al. „Poison frogs! targeted clean-label poisoning attacks on neural networks“. In: *arXiv preprint arXiv:1804.00792* (2018) (cit. on p. 18).

[SS15] Reza Shokri and Vitaly Shmatikov. „Privacy-Preserving Deep Learning“. In: *ACM SIGSAC Conference on Computer and Communications Security, 2015*. 2015, pp. 1310–1321 (cit. on pp. 1, 9, 10, 16, 63).

[STS16] Shiqi Shen, Shruti Tople, and Prateek Saxena. „Auror: Defending against poisoning attacks in collaborative deep learning systems“. In: *Proceedings of the 32nd Annual Conference on Computer Security Applications*. 2016, pp. 508–519 (cit. on pp. 18, 20).

[Sza01] Paweł J. Szabłowski. „Discrete Normal distribution and its relationship with Jacobi Theta functions“. In: *Statistics & Probability Letters* 52.3 (2001), pp. 289–299 (cit. on p. 75).

[Sze+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, et al. „Intriguing properties of neural networks“. In: *arXiv preprint arXiv:1312.6199* (2013) (cit. on p. 19).

[Tay20] Robert Taylor. *Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) algorithm implementation*. `https://bitbucket.org/rtaylor/pylbfgs/src/master/`. 2020 (cit. on pp. 12, 26).

[TP12] Amin Tavakoli and Ali Pourmohammad. „Image denoising based on compressed sensing“. In: *International Journal of Computer Theory and Engineering* 4.2 (2012), p. 266 (cit. on p. 17).

[Tru+19] Stacey Truex, Nathalie Baracaldo, Ali Anwar, et al. „A Hybrid Approach to Privacy-Preserving Federated Learning“. In: *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. AISec'19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 1–11 (cit. on p. 36).

[TTGL20] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. „Data poisoning attacks against federated learning systems“. In: *European Symposium on Research in Computer Security*. Springer. 2020, pp. 480–501 (cit. on pp. 18–20).

[VC02] Jaideep Vaidya and Chris Clifton. „Privacy preserving association rule mining in vertically partitioned data“. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2002, pp. 639–644 (cit. on p. 10).

[Wan+18] Hongyi Wang, Scott Sievert, Shengchao Liu, et al. „ATOMO: Communication-efficient Learning via Atomic Sparsification“. In: *NeurIPS*. 2018 (cit. on p. 17).

[Wan+19] Bolun Wang, Yuanshun Yao, Shawn Shan, et al. „Neural cleanse: Identifying and mitigating backdoor attacks in neural networks“. In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 707–723 (cit. on p. 20).

[Wen+17] Wei Wen, Cong Xu, Feng Yan, et al. „TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning“. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 1508–1518 (cit. on p. 17).

[Wu+17] Xi Wu, Fengan Li, Arun Kumar, et al. „Bolt-on Differential Privacy for Scalable Stochastic Gradient Descent-Based Analytics“. In: *Proceedings of the 2017 ACM International Conference on Management of Data*. SIGMOD '17. Chicago, Illinois, USA: Association for Computing Machinery, 2017, pp. 1307–1322 (cit. on p. 16).

[Xia+15] Huang Xiao, Battista Biggio, Gavin Brown, et al. „Is feature selection secure against training data poisoning?“ In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1689–1698 (cit. on p. 18).

[XKG18] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. „Generalized byzantine-tolerant sgd“. In: *arXiv preprint arXiv:1802.10116* (2018) (cit. on p. 20).

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. „Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms“. In: *CoRR* abs/1708.07747 (2017). arXiv: 1708.07747 (cit. on p. 79).

[YBS20] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. „Salvaging federated learning by local adaptation“. In: *arXiv preprint arXiv:2002.04758* (2020) (cit. on p. 17).

[YCKB18] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. „Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates“. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 5650–5659 (cit. on pp. 19, 20).

[YLCT19] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. „Federated machine learning: Concept and applications“. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19 (cit. on p. 10).

[YZT14] Lean Yu, Yang Zhao, and Ling Tang. „A compressed sensing based AI learning paradigm for crude oil price forecasting“. In: *Energy Economics* 46 (2014), pp. 236–245 (cit. on p. 17).

[Zhu+20] Yuqing Zhu, Xiang Yu, Yi-Hsuan Tsai, et al. *Voting-based Approaches For Differentially Private Federated Learning*. 2020. arXiv: 2010.04851 [cs.LG] (cit. on pp. 41–43, 47).

[ZLH19] Ligeng Zhu, Zhijian Liu, and Song Han. „Deep Leakage from Gradients“. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, et al. 2019, pp. 14747–14756 (cit. on pp. 2, 15, 35).

[ZMB20] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. „iDLG: Improved Deep Leakage from Gradients“. In: (2020) (cit. on pp. 2, 15, 35).

[ZWZZC19] Lingchen Zhao, Qian Wang, Qin Zou, Yan Zhang, and Yanjiao Chen. *Privacy-Preserving Collaborative Deep Learning with Unreliable Participants*. 2019. arXiv: 1812.10113 [cs.CR] (cit. on p. 16).

[ZZXYW12] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. „Functional Mechanism: Regression Analysis under Differential Privacy“. In: *Proc. VLDB Endow.* 5.11 (July 2012), pp. 1364–1375 (cit. on p. 16).

Webseiten

[Dal21] Sam Daley. *23 EXAMPLES OF ARTIFICIAL INTELLIGENCE SHAKING UP BUSINESS AS USUAL*. 1-04-2021. URL: https://builtin.com/artificial-intelligence/examples-ai-in-industry (cit. on p. 2).

[For18] World Economic Forum. *Harnessing Artificial Intelligence for the Earth (Fourth Industrial Revolution for the Earth Series)*. 2018. URL: http://www3.weforum.org/docs/Harnessing_Artificial_Intelligence_for_the_Earth_report_2018.pdf (cit. on p. 1).

[Lab17] Mate Labs. *How these researchers tried something unconventional to come out with a smaller yet better Image Recognition.* May 2017. URL: https://medium.com/@matelabs_ai/how-these-researchers-tried-something-unconventional-to-came-out-with-a-smaller-yet-better-image-544327f30e72#.i3227lyhb (cit. on p. 78).

[OWK21] INC OWKIN. *OWKIN FEDERATED LEARNING TO ACCELERATE MEDICAL RESEARCH*. 2021. URL: https://owkin.com/ (cit. on p. 2).

[PC16] REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT and OF THE COUNCIL. *On the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. 27-04-2016. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN (cit. on p. 1).

[PC21] REGULATION OF THE EUROPEAN PARLIAMENT and OF THE COUNCIL. *LAYING DOWN HARMONISED RULES ON ARTIFICIAL INTELLIGENCE (ARTIFICIAL INTELLIGENCE ACT) AND AMENDING CERTAIN UNION LEGISLATIVE ACTS*. 21-04-2021. URL: https://ec.europa.eu/newsroom/dae/items/709090 (cit. on p. 1).

[Uni12] The European Union. *CHARTER OF FUNDAMENTAL RIGHTS OF THE EUROPEAN UNION*. 26-10-2012. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:12012P/TXT&from=EN (cit. on p. 1).

[Uni21] Yale University. *Lack of ICU beds tied to thousands of excess COVID-19 deaths, study finds*. 2.02.2021. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN (cit. on p. 1).

[Wik] Wikipedia. *Cambridge Analytica*. URL: https://en.wikipedia.org/wiki/Cambridge_Analytica (cit. on p. 1).

8 Appendix

8.1 Proofs for FL-SIGN and FL-SIGN-DP

8.1.1 Proof of Lemma 1

Proof. We first show that $Z/\sqrt{2\pi}\xi \leq 1 + \frac{2e^{-2\pi^2\xi^2}}{1-e^{-6\pi^2\xi^2}}$, where $Z = \sum_{x\in\mathbb{Z}} \exp(-(x-\mu)^2/2\xi^2)$, which implies the upper bound. From [Sza01], $Z = \sqrt{2\pi}\xi\vartheta_3(\pi\mu, \exp(-2\pi^2\xi^2))$, where $\vartheta_3(u,r) = 1 + 2\sum_{i\geq 1} r^{i^2}\cos(2iu)$ is a Jacobi Theta function. Then,

$$\begin{aligned} 1 + 2\sum_{i\geq 1} r^{i^2}\cos(2iu) &\leq 1 + 2r\sum_{i\geq 0} r^{3i} \\ &\leq 1 + \frac{2r}{1-r^3} \end{aligned}$$

if $|r| < 1$. The lower bound can be derived similarly using the fact that $\cos(2iu) \geq -1$. □

8.1.2 Proof of Theorem 2

Proof. Without loss of generality, suppose that $\mathcal{DG}_\xi : \mathbb{R} \to \mathbb{Z}^n$.

We apply the moments accountant [Aba+16] and show that $\alpha_{\mathcal{DG}}(\lambda)$ can be upper bounded efficiently without evaluating the pmf of $\mathcal{DG}$.

Let $\eta_0^{\mathcal{DG}}(x|\xi) = \mathsf{pmf}_{\mathcal{DG}(0,\xi)}(x)$ and $\eta_1^{\mathcal{DG}}(x|\xi) = (1-q)\mathsf{pmf}_{\mathcal{DG}(0,\xi)}(x) + q\mathsf{pmf}_{\mathcal{DG}(1,\xi)}(x)$ where $\mathsf{pmf}_{\mathcal{DG}(\mu,\xi)}(x) = \Pr_{x\sim\mathcal{DG}(\mu,\xi)}[x]$. Then,

$$\alpha_{\mathcal{DG}}(\lambda) = \log\max(E_1'(\lambda,\xi), E_2'(\lambda,\xi))$$

where

$$\begin{aligned} E_1'(\lambda,\xi) &= \sum_{x=-\infty}^{\infty} \eta_0^{\mathcal{DG}}(x|\xi)\cdot\left(\frac{\eta_0^{\mathcal{DG}}(x|\xi)}{\eta_1^{\mathcal{DG}}(x|\xi)}\right)^\lambda \\ &\leq \frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}} \sum_{x=-\infty}^{\infty} \eta_0^{\mathcal{G}}(x|\xi)\cdot\left(\frac{\eta_0^{\mathcal{G}}(x|\xi)}{\eta_1^{\mathcal{G}}(x|\xi)}\right)^\lambda \\ &\leq \frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}} \int_{\mathbb{R}} \eta_0^{\mathcal{G}}(y|\xi)\cdot\left(\frac{\eta_0^{\mathcal{G}}(y|\xi)}{\eta_1^{\mathcal{G}}(y|\xi)}\right)^\lambda dy \\ &= \frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}} E_1(\lambda,\xi) \end{aligned}$$

where the first inequality follows from Lemma 1. Using a similar reasoning we obtain that

$$E_2'(\lambda,\xi) \leq \frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}} E_2(\lambda,\xi)$$

The theorem follows from Theorem 2 in [Aba+16]. □

8.1.3 Proof of Theorem 3

As opposed to the continuous case, the sum of discrete Gaussian random variables $\sum_i \mathcal{DG}(\mu_i,\xi_i)$ does not follow the distribution of $\mathcal{DG}(\sum_i \mu_i, \sqrt{\sum_i \xi_i^2})$, though it is very close to that if ξ_i is sufficiently large. The exact difference is quantified by the following Lemma from [MW17].

Lemma 3 ([MW17], Theorem 2.1). *If $\xi_i \geq \sqrt{\ln(2+2/\nu)}/\pi$ and $\boldsymbol{\mu}_i \in \mathbb{R}^n$, then*

$$\frac{1-\nu}{1+\nu} \leq \frac{\Pr_{\mathbf{z}\sim\sum_i \mathcal{DG}(\boldsymbol{\mu}_i,\xi_i)}[\mathbf{z}]}{\Pr_{\mathbf{z}\sim\mathcal{DG}\left(\sum_i \boldsymbol{\mu}_i, \sqrt{\sum_i \xi_i^2}\right)}[\mathbf{z}]} \leq \frac{1+\nu}{1-\nu}$$

for any $\mathbf{z} \in \mathbb{Z}^n$

Intuitively, if $Z = \sum_{x\in\mathbb{Z}} \exp(-(x-\mu)^2/2\xi_i^2) \approx \sqrt{2\pi}\xi_i$ then $\mathcal{DG}(\mu_i,\xi_i) \approx \mathcal{G}(\mu_i,\xi_i)$, in which case $\sum_i \mathcal{DG}(\mu_i,\xi_i) \approx \sum_i \mathcal{G}(\mu_i,\xi_i) = \mathcal{G}(\mu_i,\sum_i \xi_i) \approx \mathcal{DG}(\mu_i,\sum_i \xi_i)$, which follows from Lemma 3. Indeed, it also follows from the proof of Lemma 1 that $Z/\sqrt{2\pi}\xi_i \leq \vartheta_3(\pi\mu, \exp(-2\pi^2\xi_i^2)) \leq 1 + \frac{2e^{-2\pi^2\xi_i^2}}{1-e^{-6\pi^2\xi_i^2}} \leq 1 + \frac{2}{\exp(2\xi_i^2\pi^2)-1} \leq 1 + \frac{2}{\exp(\xi_i^2\pi^2)-2} \leq 1+\nu$ which provides some insight into the condition on ξ_i in Lemma 3.

For example, $\nu < 10^{-4}$ is satisfied if $\xi_i > 1$.

Let $\widehat{\mathcal{DG}}_\xi$ denote the distributed Gaussian mechanism which returns $\sum_{k=1}^N \mathcal{DG}(\boldsymbol{\mu}_k, \xi/\sqrt{|\mathbb{K}|})$ where $\boldsymbol{\mu}_k \in \mathbb{R}^n$. The next lemma, which directly follows from Theorem 2 and Lemma 3, implies Theorem 3.

Lemma 4. *If $\xi \geq \sqrt{|\mathbb{K}|\ln(2+2/\nu)}/\pi$, then $\alpha_{\widehat{\mathcal{DG}}}(\lambda|q) \leq \alpha_{\mathcal{G}}(\lambda|q) + \log\left(\frac{(1+\kappa(\xi))^\lambda}{(1-\kappa(\xi))^{\lambda+1}}\left(\frac{1+\nu}{1-\nu}\right)^3\right)$.*

8.1.4 Convergence Proofs

All the proofs are simple adaptations of Theorem 2 from [BZAA18]. Here we outline only the main deviations from the proof of that theorem.

Assumptions:

1. *Lower bound:* For all x and some constant f^*, $f(x) \geq f^*$, where f denotes the loss/objective function.

2. *Smoothness:* Let $g(x)$ denote the gradient of the objective function f evaluated at x. Then, for all x, y and some non-negative constant $\mathbf{L} = (L_1, L_2, \ldots, L_n)$,

$$|f(y) - [f(x) + g(x)^\mathsf{T}(y-x)]| \leq 1/2 \sum_i L_i (y_i - x_i)^2$$

3. *Variance bound:* Upon receiving query $x \in \mathbb{R}^n$, the stochastic gradient oracle gives us an independent, unbiased estimate $\hat{g}$ that has bounded variance per coordinate: $\mathbb{E}[\hat{g}(x)] = g(x)$, $\mathbb{E}[(\hat{g}(x)_i - g(x)_i)^2] \leq \tau_i^2$ for a vector of non-negative constants $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_n)$.

4. *Unimodal, symmetric gradient noise:* At any given point x, each component of the stochastic gradient vector $\hat{g}(x)$ has unimodal distribution that is also symmetric about the mean.

Note that adding extra Gaussian noise to each gradient component for the purpose of differential privacy will not violate Assumption 4.

Theorem 6. *If $|\mathbb{B}| = T_{\mathsf{cl}}$, $T_{\mathsf{gd}} = 1$, and $\gamma = \sqrt{\frac{f_0 - f_*}{||\mathbf{L}||_1 T_{\mathsf{cl}}}}$, then*

1. *For FL-SIGN in the Random Update Attack,*

$$\frac{1}{T_{\mathsf{cl}}} \sum_{t=0}^{T_{\mathsf{cl}}-1} \mathbb{E}||g_t||_1 \leq \frac{2}{\sqrt{T_{\mathsf{cl}}}}\left(\frac{\sqrt{2}||\boldsymbol{\tau}||_1}{(1-\alpha)\sqrt{CN}} + \sqrt{||\mathbf{L}||_1(f_0 - f^*)}\right)$$

   where α denotes the fraction malicious clients and $|g_i|/\tau_i \leq 2/\sqrt{3}$ for all $1 \leq i \leq n$.

2. *For FL-SIGN-DP,*

$$\frac{1}{T_{\mathsf{cl}}} \sum_{t=0}^{T_{\mathsf{cl}}-1} \mathbb{E}||g_t||_1 \leq \frac{2}{\sqrt{T_{\mathsf{cl}}}}\left(\frac{||\boldsymbol{\tau}||_1}{\sqrt{CN}} + \frac{\sqrt{3n}\sigma||\boldsymbol{\tau}||_1}{CN} + \sqrt{||\mathbf{L}||_1(f_0 - f^*)}\right)$$

   if $|g_i|/\tau_i \leq 2/\sqrt{3}$ for all $1 \leq i \leq n$.

Proof. The primary focus of all the proofs is to bound the probability that a client computes the sign of a parameter update correctly. Let $M = CN$. As in [BZAA18], let $Z_i \in [0, M]$ denote the number of correct sign bits received by the aggregator for parameter i, and p denotes the probability that a honest client computes the correct bit. Let $\omega = p - \frac{1}{2}$.

1. **Random Update Attack:** Notice that the probability that a sign of any parameter is correct at a malicious client is $1/2$, and each client acts independently from each other. Hence, $\mathrm{E}[Z_i] = (1-\alpha)Mp + \frac{1}{2}\alpha M$ and $\mathrm{Var}[Z] = \frac{1}{4}\alpha M + (1-\alpha)Mp(1-p)$. The probability that a vote fails for the i^{th} parameter is identical to $\mathrm{P}[Z_i \leq M/2]$, which, likewise in [BZAA18], can be bounded as follows.

$$
\begin{aligned}
\mathrm{P}[Z_i \leq M/2] &= \mathrm{P}[\mathrm{E}[Z_i] - Z_i \geq \mathrm{E}[Z_i] - M/2] \\
&\leq \frac{1}{1 + \frac{(\mathrm{E}[Z_i]-N/2)^2}{\mathrm{Var}[Z_i]}} && \text{(by Cantelli's inequality)} \\
&\leq \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Z_i]}{(\mathrm{E}[Z_i] - M/2)^2}} && \text{(by } 1 + x^2 \geq 2x\text{)} \\
&\leq \frac{1}{2\sqrt{M}}\sqrt{\frac{\frac{1}{4}\alpha + (1-\alpha)p(1-p)}{(1-\alpha)^2(p-\frac{1}{2})^2}} \\
&\leq \frac{1}{2\sqrt{M}}\sqrt{\frac{\frac{1}{4}\alpha}{(1-\alpha)^2(p-\frac{1}{2})^2}} \\
&+ \frac{1}{2\sqrt{M}}\sqrt{\frac{p(1-p)}{(1-\alpha)(p-\frac{1}{2})^2}} \\
&\leq \frac{\sqrt{\alpha}}{4\sqrt{M}(1-\alpha)|\omega|} + \frac{1}{2\sqrt{M}}\sqrt{\frac{\frac{1}{4} - \omega^2}{(1-\alpha)\omega^2}} \\
&\leq \frac{\sqrt{3\alpha}\tau_i}{2\sqrt{M}(1-\alpha)|g_i|} + \frac{\tau_i}{\sqrt{M(1-\alpha)}|g_i|} && (8.1) \\
&\leq \frac{\tau_i(\sqrt{\alpha} + \sqrt{1-\alpha})}{\sqrt{M}(1-\alpha)|g_i|} \\
&\leq \frac{\sqrt{2}\tau_i}{\sqrt{M(1-\alpha)}|g_i|}
\end{aligned}
$$

   where, in Eq. (8.1), we used that $\frac{1}{4\omega^2} - 1 \leq 4\tau_i^2/g_i^2$ and $1/|\omega| \leq 2\sqrt{3}\tau_i/|g_i|$ for $|g_i|/\tau_i < 2/\sqrt{3}$ based on Lemma 1 in [BZAA18]. The rest of the derivation is identical to the proof of Theorem 2 in [BZAA18].

2. **FL-SIGN-DP:** The Gaussian noise is added to the sum of signs. Let Y_i denote the random variable describing the noise added by the clients to Z_i.

$$
\begin{aligned}
\mathrm{P}[Z_i + Y_i \leq M/2] &\leq \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Z_i + Y_i]}{(\mathrm{E}[Z_i + Y_i] - M/2)^2}} \\
&\leq \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Z_i] + \mathrm{Var}[Y_i]}{(\mathrm{E}[Z_i] - M/2)^2}} && \text{(by independence and } \mathrm{E}[Y_i] = 0\text{)} \\
&\leq \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Z_i]}{(\mathrm{E}[Z_i] - M/2)^2}} \\
&+ \frac{1}{2}\sqrt{\frac{\mathrm{Var}[Y_i]}{(\mathrm{E}[Z_i] - M/2)^2}} && (8.2)
\end{aligned}
$$

Based on [BZAA18],

$$\begin{aligned}\frac{1}{2}\sqrt{\frac{\mathrm{Var}\left[Z_i\right]}{(\mathrm{E}\left[Z_i\right]-M/2)^2}} &\leq \frac{1}{2}\sqrt{\frac{M\left(\frac{1}{4}-\omega^2\right)}{M^2\omega^2}} \\ &\leq \frac{1}{2}\sqrt{\frac{M4\tau_i/|g_i|}{M^2\omega^2}} \\ &\leq \frac{\tau_i}{\sqrt{M}|g_i|} \end{aligned} \tag{8.3}$$

Moreover, if $|g_i|/\tau_i \leq 2/\sqrt{3}$, then $1/\omega^2 \leq 12\tau_i^2/g_i^2$, and hence

$$\begin{aligned}\frac{1}{2}\sqrt{\frac{\mathrm{Var}\left[Y_i\right]}{(\mathrm{E}\left[Z_i\right]-M/2)^2}} &\leq \frac{1}{2}\sqrt{\frac{n\sigma^2}{M^2\omega^2}} \\ &\leq \frac{1}{2}\sqrt{\frac{12n\sigma^2\tau_i^2/g_i^2}{M^2}} \\ &\leq \frac{\sqrt{3}\sqrt{n}\sigma\tau_i}{M|g_i|} \end{aligned} \tag{8.4}$$

Plugging Eq. (8.3) and (8.4) into Eq. (8.2), we obtain that the probability that the noisy vote fails for the i^{th} coordinate is bounded as

$$\mathrm{P}\left[Z_i + Y_i \leq M/2\right] \leq \frac{\tau_i}{\sqrt{M}|g_i|} + \frac{\sqrt{3}\sqrt{n}\sigma\tau_i}{M|g_i|}$$

if $|g_i|/\tau_i \leq 2/\sqrt{3}$. The claim follows from the proof of Theorem 2 in [BZAA18].

□

8.2 Selection of Hyperparameters

Strictly speaking, the selection of hyperparameters in each private schemes, such as batch size $|\mathbb{B}|$, scaling factor γ, or sensitivity S, must also be differentially private. One option is to use public data for this purpose which comes from the same distribution as the clients' private training data. The selection of hyperparameters can also be performed using more sophisticated methods like the one in Appendix D of [Aba+16].

8.3 Experimental Set-up

This section describes the experimental set-up that are used to evaluate the accuracy of our proposal. The following datasets were used: MNIST, Fashion-MNIST, IMDB, LFW and CIFAR.

8.3.1 Model Architectures:

- For MNIST and Fashion-MNIST, we use a model [MMRHA16] with the following architecture: a convolutional neural network (CNN) with two 5x5 convolution layers (the first with 32 filters, the second with 64, each followed with 2x2 max pooling), a fully connected layer with 512 units and ReLu activation, and a final softmax output layer. This results in 1,663,370 parameters in total.

- The LFW dataset is used with a CNN of three 3x3 convolution layers (32, 64, and 128 filters, each followed with 2x2 max pooling), a fully connected layer with 256 units and ReLU activation, and a final softmax output layer with 2 units. To test the property inference attack from [MSDS19], batch size is set to 32, and the SGD learning rate is 0.01.

- The model that we use for the CIFAR dataset is called "All-CNN-C" in [SDBR15] [Lab17], which consists of a CNN of 3 blocks: the first block has three 3x3 convolutions layers with 96 filters (the last layer has a strides of 2x2 and dropout of 0.5 is applied), the ReLu activation is used per layer. The second block has the same configuration as the previous block, except the filter size which is 192 for each layer. The last block has one 3x3 convolutions layer with 192 filters,

followed by two 1x1 convolution layers: the first with 192 filters (Relu activation) and the second with 10 filters. The last layer is connected with a global average pooling layer and uses softmax activation. We use also the Adam optimizer with a learning rate of 0.001. This results in 1,369,738 parameters in total.

- We use the following model for the IMDB dataset: one embedding layer with an output size of 50 (the vocabulary size is set to 5000 and the maximum length input to 400), followed by a convolution layer of one dimension with a kernel size of 5 and 250 filters; and a max pooling layer of size 3; followed by a LSTM layer with an output size set to 70 and an output layer with one unit that uses a sigmoid activation function. We use the Adam optimizer with a learning rate of 0.001. This results in 402,701 parameters in total.

- For our medical dataset, as in [Ava+18], we use a fully connected neural network model with the following architecture: two hidden layers of 200 units, which use a Relu activation function followed by an output layer of 1 unit with sigmoid activation function and a binary cross entropy loss function. A dropout layer with a rate set to 0.5 is used between each hidden layer and between the last hidden layer and the output layer. This results in 1,496,601 parameters in total. We tune η from 0.01 to 0.5 with an increment value of 0.005.

8.3.2 Datasets

The following datasets were used:

- The MNIST database of handwritten digits. It consists of 28 x 28 grayscale images of digit items and has 10 output classes. The training set contains 60,000 data samples while the test/validation set has 10,000 samples [LC10] [Cho+15b].

- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We augment the dataset to 500,000 training images by randomly shifting the original images horizontally and vertically and by randomly flipping the original images horizontally [KNH] [Cho+15b].

- Fashion-MNIST database of fashion articles consists of 60,000 28x28 grayscale images of 10 fashion categories, along with a test set of 10,000 images [XRV17] [Cho+15b].

- IMDB Movie reviews sentiment classification dataset of 25,000 movies reviews, labeled by sentiment (positive/negative) [Cho+15b]. The test set contains also 25,000 movies reviews.

- Labeled Faces in the Wild (LFW) dataset: consists of 13,000 $62 \cdot 47$ RGB images of faces collected from the web [HRBL07]. The pixel of each image is an unsigned integer in the range between 0 and 255. We rescale themto the range [0,1] instead. And we did exactly the same with MNIST, Fashion-MNIST and CIFAR-10.

- As medical dataset, we used EHR data from the Premier healthcare database[1] which is one of the largest clinical databases in the United States, collecting information from millions of patients over a period of 12 months from 415 hospitals in the USA [FGLB18]. These hospitals are supposedly representative of the United States hospital experience [FGLB18]. Each hospital in the database provides discharge files that are dated records of all billable items (including therapeutic and diagnostic procedures, medication, and laboratory usage) which are all linked to a given patient's admission [FGLB18; MR14]. The initial snapshot of the database used in our work (before pre-processing step) comprises the EHR data of 1,271,733 hospital admissions. Electronic Health Record (EHR) is a digital version of a patient's paper chart readily available in hospitals. For developing supervised learning and specifically deep learning models, we focus on a specific set of features from EHR data. The features of interest that capture the patients information are summarized in Table 8.6. There is a total of 24,428 features per patient, mainly

[1] https://www.premierinc.com/newsroom/education/premier-healthcare-database-whitepaper

due to the variety of drugs possibly served. As in [Ava+18], we also removed all the features which appear on less than 100 patients' records, hence, the number of features was reduced to 7,280 features. The Medication regimen complexity index (MRCI) [Mcd+12] is an aggregate score computed from a total of 65 items, whose purpose is to indicate the complexity of the patient's situation. The minimum MRCI score for a patient is 1.5, which represents a single tablet or capsule taken once a day as needed (single medication). However the maximum is not defined since the number of medications increases the score [Mcd+12]. In our case, after statistical analysis of our dataset, we consider the MRCI score as ranging from 2 to 60. Most real datasets like ours are generally imbalanced with a skewed distribution between the classes. In our case, the positive cases (patients who die during their hospital stay) represent only 3% of all patients. Table 8.7 gives more details about this distribution after the pre-processing step which is discussed in Appendix 8.4.1. To deal with this well-known problem, we have decided to use the downsampling technique [Mor16; HG09], a standard solution used for this purpose.[2]

The In-hospital Mortality Prediction Scenario: The ability to accurately predict the risks in the patient's perspectives of evolution is a crucial prerequisite in order to adapt the care that certain patients receive [FGLB18]. We consider the scenario where several hospitals are collaborating to train models for in-hospital mortality prediction using our Federated Learning schemes. This well-studied real-world problem consists in trying to precisely identify the patients who are at risk of dying from complications during their hospital stay [Ava+18; Ra18; FGLB18]. As commonly found in the literature [FGLB18], for such predictions, we focus on hospital admissions of adults hospitalized for at least 3 days, excluding elective admissions.

8.4 Medical Data: Data Pre-Processing & Experimental Set-up Details

This section describes the experimental setting which is used to evaluate the accuracy and the privacy of our proposals on the Medical dataset.

8.4.1 Medical Data: Data Pre-Processing

1. **Features normalization**: we extract from the dataset the values of each feature represented in Table 8.6. For gender, we use one-hot encoding: Male, Female and Unknown. Similarly, for admission type we use 4 features: Emergency, Urgent, Trauma Center, and Unknown [3]. For drugs and ICD9 codes, we extract 24,419 features which correspond to the different drugs (name and dosage), procedures codes and diagnosis codes. As in [Ava+18], we also removed all the features which appear on less than 100 patients' records, hence, the number of features was reduced to 7,280 features. A given patient receives only a few of the possible drugs served, resulting in a very sparse patient's record. We use a MinMax normalization for age and MRCI in order to rescale the values of these features between 0 and 1 (using MinMaxScaler class of scikit-learn[4]). The labels that we consider are boolean: true means that the patient died during his hospital stay while false means she survived.

2. **Patients filtering**: We consider patient and drug information of the first day at the hospital so that we can make predictions 24 hours after admission (as commonly found in the literature [Ra18; FGLB18]). We filter out the pregnant and new-born patients because the medication types and admission services are not the same for theses two categories of patients. Our model prediction is built without patients' historical medical data. This has the advantage to require minimum patient's information and to work for new patients.

3. **Hospitals filtering**: The dataset contains 415 hospitals for a total size of 1,271,733 records. We split randomly the dataset into disjoint training and testing data (80% and 20% respectively). The final dataset for testing contains 254,347 patients, with 7,882 deceased patients and 246,465 non-deceased patients (see Table 8.7).

[2] We have also tested weighted loss function and oversampling techniques. Experimentally we observed that downsampling outperforms the other techniques whatever the considered scheme.

[3] `https://www.resdac.org/cms-data/variables/claim-inpatient-admission-type-code-ffs`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`

Using Client-Level differential privacy requires to add more noise than Record-Level differential privacy, because the privacy purposes are not the same as detailled in Section 2.2. To reduce the noise (when ϵ is fixed) and then improve the utility, we have to reduce the number of iterations or to reduce the sampling probability which are the parameters used to compute ϵ. We therefore have two options to reduce the sampling probability:

- Reducing the number of clients selected at each round $|\mathbb{K}|$. However this option also decreases the amount of data, and hence have a negative impact on the utility. We therefore preferred to use the next option.
- Increasing the total number of clients N: we created more hospitals by splitting randomly the training data over 5011 "virtual" hospitals. We also, took care to have at least one in-hospital dead patient per hospital. Each hospital contains 203 patients except one which has 356 patients. We created 5011 hospitals [5] in order to have approximately the same number of patients per hospital, each of them with some in-hospital dead patients.

  In practise, Client-Level differential privacy is more adapted to an environment with a large set of clients as explained in [MRTZ18; GKN17].

8.4.2 Medical Data: Performance Metrics

We use the following metrics:

- *Balanced accuracy* [BOSB10; BDA13] is computed as $1/2 \cdot (\frac{TP}{P} + \frac{TN}{N}) = \frac{TPR + TNR}{2}$ and is mainly used with imbalanced data. *True Positive Rate* (*TPR*) and *True Negative Rate* (*TNR*): $TPR = \frac{TP}{P}$ and $TNR = \frac{TN}{N}$, where P and N are the number of positive and negative instances, respectively, and TP and TN are the number of true positive and true negative instances. We note that traditional ("non-balanced") accuracy metrics such as $\frac{TP+TN}{P+N}$ can be misleading for very imbalanced data [Ako17]: in our dataset, the minority class has only 3% of all the training samples (see Table 8.7), which means that a biased (and totally useless) model always predicting the majority class would have a (non-balanced) accuracy of 97%.
- The *area under the ROC curve* (*AUROC*) is also a frequently used accuracy metric. The ROC curve is calculated by varying the prediction threshold from 1 to 0, when *TPR* and *FPR* are calculated at each threshold. The area under this curve is then used to measure the quality of the predictions. A random guess has an *AUROC* value of 0.5, whereas a perfect prediction has the largest *AUROC* value of 1.

8.4.3 Evaluation Method.

First, we split randomly the dataset of each hospital into disjoint training and testing data (80% and 20% respectively). An entire federated run is executed with this split, and all the metrics are evaluated in every round on the union of all clients' testing data. All metric values of the round with the best balanced metric are recorded.

Tab. 8.1: Common environment and configuration of FL-SIGN, FL-CS, FL-TOP and FL-STD. $\gamma = 0.001$.

Datasets	MNIST Fashion-MNIST	IMDB	CIFAR
Parameters	$N = 1000$; $C = 0.1$; $\|D_k\| = 60$; $\|\mathbb{B}\| = 10$; $T_{\text{gd}} = 30$; $T_{\text{cl}} = 100$; SGD $(\eta = 0.215)$	$N = 1000$; $C = 0.1$; $\|D_k\| = 25$; $\|\mathbb{B}\| = 25$; $T_{\text{gd}} = 5$; $T_{\text{cl}} = 100$; $ADAM$ $(\eta = 0.001)$	$N = 1000$; $C = 0.1$; $\|D_k\| = 500$; $\|\mathbb{B}\| = 50$; $T_{\text{gd}} = 50$; $T_{\text{cl}} = 400$; $ADAM$ $(\eta = 0.001)$

[5] We consider 5010 hospitals with FL-TOP and FL-TOP-DP, instead of 5011. The dataset of the non-selected hospital is used as public data to fix the updatable weights at the server side.

Tab. 8.2: Common environment of the privacy part. $\gamma = 0.005$ and $T_{\text{cl}} = 200$. For FL-STD-DP, S is set to 1.73 and 2.15 for MNIST and Fashion-MNIST, respectively. For FL-SIGN-DP, S is fixed to $\sqrt{n}$.

Algorithms \ Datasets	MNIST & Fashion-MNIST
FL-SIGN-DP & FL-STD-DP	$N = 6000$; $C = 1/60$; $\lvert D_k \rvert = 10$; $\lvert \mathbb{B} \rvert = 10$; $T_{\text{gd}} = 5$; $SGD(\eta = 0.215)$; $n = 1,663,370$; $\delta = 10^{-5}$

Tab. 8.3: Parameter Configuration for the Backdoor Attacks. FL-SIGN is used with the vote aggregation $\gamma = 0.001$.

Attacks \ Datasets	MNIST
In-backdoor	$N = 10$; $C = 1$; $\lvert D_k \rvert = 6000$; $\lvert \mathbb{B} \rvert = 10$; $T_{\text{gd}} = 3000$; $T_{\text{cl}} = 40$; $SGD(\eta = 0.215)$
Out-backdoor	$N = 10$; $C = 1$; $\lvert D_k \rvert = 6000$; $\lvert \mathbb{B} \rvert = 10$; $T_{\text{gd}} = 3000$; $T_{\text{cl}} = 100$; $SGD(\eta = 0.215)$
Attacks \ Datasets	CIFAR
In-backdoor	$N = 10$; $C = 1$; $\lvert D_k \rvert = 50000$; $\lvert \mathbb{B} \rvert = 100$; $T_{\text{gd}} = 1000$; $T_{\text{cl}} = 100$; $ADAM(\eta = 0.001)$
Out-backdoor	$N = 10$; $C = 1$; $\lvert D_k \rvert = 50000$; $\lvert \mathbb{B} \rvert = 100$; $T_{\text{gd}} = 1000$; $T_{\text{cl}} = 300$; $ADAM(\eta = 0.001)$

8.5 Computational Environment

Our experiments were performed on a server running Ubuntu 18.04 LTS equipped with an Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz, 192GB RAM, and two NVIDIA Quadro P5000 GPU card of 16 Go each. We use Keras 2.2.0 [Cho+15a] with a TensorFlow backend 1.12.0 [**TensorFlow**] and Numpy 1.14.3 [Oli06] to implement our models and experiments. We use Python 3.6.5 and our code runs on a Docker container to simplify the reproducibility.

Algos	*Parameters*
FL-STD & FL-STD-DP (r=1.0)	$S = 2.15$; $C = 1/60$; $N = 6000$; $T_{\text{cl}} = 200$; $T_{\text{gd}} = 5$; $\lvert \mathbb{B} \rvert = 10$; $\lvert D_k \rvert = 10$; $n = 1,663,370$; $\delta = 10^{-5}$; $SGD(\eta = 0.215)$; $\sigma = 1.54$
FL-CS,FL-RND,FL-FREQ and their private extensions (r=0.2)	$S = 0.98$; $C = 1/60$; $N = 6000$; $T_{\text{cl}} = 200$; $T_{\text{gd}} = 5$; $\lvert \mathbb{B} \rvert = 10$; $\lambda = 10^{-5}$; $\lvert D_k \rvert = 10$; $n = 1,663,370$; $\delta = 10^{-5}$; $SGD(\eta = 0.215)$; $\eta_G = 0.35$; $\rho = 0.9$; $P = 200$; $\sigma = 1.54$
FL-CS,FL-RND,FL-FREQ and their private extensions (r=0.1)	$S = 0.69$; $C = 1/60$; $N = 6000$; $T_{\text{cl}} = 200$; $T_{\text{gd}} = 5$; $\lvert \mathbb{B} \rvert = 10$; $\lambda = 10^{-5}$; $\lvert D_k \rvert = 10$; $n = 1,663,370$; $\delta = 10^{-5}$; $SGD(\eta = 0.215)$; $\eta_G = 0.35$; $\rho = 0.9$; $P = 200$; $\sigma = 1.54$
FL-CS,FL-RND,FL-FREQ and their private extensions (r=0.05)	$S = 0.47$; $C = 1/60$; $N = 6000$; $T_{\text{cl}} = 200$; $T_{\text{gd}} = 5$; $\lvert \mathbb{B} \rvert = 10$; $\lambda = 10^{-5}$; $\lvert D_k \rvert = 10$; $n = 1,663,370$; $\delta = 10^{-5}$; $SGD(\eta = 0.215)$; $\eta_G = 0.35$; $\rho = 0.9$; $P = 200$; $\sigma = 1.54$

Tab. 8.4: Common environment between the schemes on Fashion-MNIST. ρ, η_G, λ and P are only used with FL-CS and FL-CS-DP.

Algos	*Parameters*
FL-STD & FL-STD-DP (r=1.0)	$S = 0.31$; $C = 100/5011$; $N = 5011$; $T_{\text{cl}} = 100$; $T_{\text{gd}} = 5$; $n = 1,496,601$; $\delta = 10^{-5}$; $SGD(\eta = 0.1)$; $\sigma = 1.49$
FL-CS,FL-RND,FL-FREQ and their private extensions (r=0.2)	$S = 0.14$; $C = 100/5011$; $N = 5011$; $T_{\text{cl}} = 100$; $T_{\text{gd}} = 5$; $\lambda = 10^{-5}$; $n = 1,496,601$; $\delta = 10^{-5}$; $SGD(\eta = 0.1)$; $\eta_G = 1.0$; $\rho = 0.9$; $P = 200$; $\sigma = 1.49$
FL-CS,FL-RND,FL-FREQ and their private extensions (r=0.1)	$S = 0.1$; $C = 100/5011$; $N = 5011$; $T_{\text{cl}} = 100$; $T_{\text{gd}} = 5$; $\lambda = 10^{-5}$; $n = 1,496,601$; $\delta = 10^{-5}$; $SGD(\eta = 0.1)$; $\eta_G = 1.0$; $\rho = 0.9$; $P = 200$; $\sigma = 1.49$
FL-CS,FL-RND,FL-FREQ and their private extensions (r=0.05)	$S = 0.07$; $C = 100/5011$; $N = 5011$; $T_{\text{cl}} = 100$; $T_{\text{gd}} = 5$; $\lambda = 10^{-5}$; $n = 1,496,601$; $\delta = 10^{-5}$; $SGD(\eta = 0.1)$; $\eta_G = 1.0$; $\rho = 0.9$; $P = 200$; $\sigma = 1.49$

Tab. 8.5: Common environment between the schemes on the Medical dataset. ρ, η_G, λ and P are only used with FL-CS and FL-CS-DP.

Tab. 8.6: Descriptions of features

Features	*Descriptions*
Age	Value in the range of 15 and 89
Gender	Male, Female or Unknown
Admission type	Emergency, Urgent, Trauma Center: visits to a trauma center/hospital or Unknown
MRCI	Medication regimen complexity index score (ranging from 2 to 60)
Drugs and ICD9 codes	Drugs given to the patient on the 1^{st} day of hospitalization. The ICD9 codes [CUA19] are composed of procedures and diagnosis codes, the first gives details about the medical procedures performed on the patient and the second about the doctor's diagnosis of the patient. There is a total of 24,419 possible drugs and ICD9 codes.

Tab. 8.7: Number of instances for our case study. The Medical dataset contains in total 1,271,733 records.

Data	Positive cases	Negative cases	Ratio	Total
Train	32,106	985,280	3.16%	1,017,386
Test	7,882	246,465	3.10%	254,347

Datasets	*Common Parameters*
Fashion-MNIST dataset	$C = 1/60$; $N = 6000$; $T_{\text{cl}} = 200$; $T_{\text{gd}} = 5$; $\lvert\mathbb{B}\rvert = 10$; $\lvert D_k\rvert = 10$; $n = 1,663,370$; $\delta = 10^{-5}$; $SGD(\eta = 0.215)$; $\eta_G = 0.35$; $\rho = 0.9$; $P = 200$; $\sigma = 1.54$; $T_{\text{init}} = 5$
Medical dataset	$C = 100/5010$; $N = 5010$; $T_{\text{cl}} = 100$; $T_{\text{gd}} = 40$; $n = 1,496,601$; $\delta = 10^{-5}$; $SGD(\eta = 0.1)$; $\eta_G = 1.0$; $\rho = 0.9$; $P = 100$; $\sigma = 1.49$; $T_{\text{init}} = 40$

Tab. 8.8: Common environment between the schemes. ρ, η_G and P are only used with FL-CS and FL-CS-DP.

Algorithms	*Compression ratio (r)*				
	0.1%	*0.5%*	*1%*	*5%*	*10%*
FL-BASIC-DP	0.05	0.12	0.16	0.34	0.45
FL-BAS-2-DP	0.07	0.16	0.23	0.52	0.75
FL-BAS-3-DP	0.05	0.11	0.16	0.33	0.44
FL-BAS-4-DP	0.06	0.15	0.21	0.51	0.74
FL-CS-DP	0.21	0.26	0.32	0.57	0.79
FL-TOP-BIS-DP	1.25	1.59	1.79	2.18	2.34
FL-TOP-DP	0.50	0.61	0.64	0.87	1.0

Tab. 8.9: Sensitivity S used for each scheme and for different compression ratio r on Fashion-MNIST. For FL-STD-DP, S is set to 2.40.

Algorithms	*Compression ratio (r)*						
	0.01%	*0.05%*	*0.1%*	*0.5%*	*1%*	*5%*	*10%*
FL-BASIC-DP	0.01	0.03	0.05	0.11	0.16	0.34	0.46
FL-BAS-2-DP	0.01	0.03	0.04	0.09	0.14	0.31	0.44
FL-BAS-3-DP	0.01	0.04	0.06	0.12	0.18	0.35	0.49
FL-BAS-4-DP	0.02	0.03	0.05	0.12	0.15	0.31	0.44
FL-CS-DP	0.002	0.005	0.006	0.01	0.02	0.04	0.06
FL-TOP-BIS-DP	0.60	0.73	0.81	1.03	1.13	1.31	1.32
FL-TOP-DP	0.23	0.46	0.59	1.03	1.18	1.31	1.32

Tab. 8.10: Sensitivity S used for each scheme and for different compression ratio r on the medical dataset. For FL-STD-DP, S is set to 1.40.

Compression ratio (r)	Algorithms	*Performance*				
		Accuracy	*Round*	*Downstream Cost* (Kilobyte)	*Upstream Cost* (Kilobyte)	ϵ
0.1%	FL-BASIC	0.14	111	12308.94	12.31	N/A
	FL-BAS-2	0.16	185	20514.9	20.51	N/A
	FL-BAS-3	0.27	200	22.17	22.17	N/A
	FL-BAS-4	0.17	200	22.17	22.17	N/A
	FL-CS	0.37	200	22178.27	22.17	N/A
	FL-TOPK-BIS	0.59	198	21.95	21.95	N/A
	FL-TOP	**0.78**	199	**22.06**	**22.06**	N/A
	FL-BASIC-DP	0.14	167	18518.85	18.51	0.95
	FL-BAS-2-DP	0.14	124	13750.53	13.75	0.88
	FL-BAS-3-DP	-	-	-	-	-
	FL-BAS-4-DP	0.15	137	15.19	15.19	0.90
	FL-CS-DP	0.36	197	21845.59	21.84	1
	FL-TOPK-BIS-DP	0.59	196	21.73	21.73	0.99
	FL-TOP-DP	**0.76**	199	**22.06**	**22.06**	**1**
0.5%	FL-BASIC	0.65	193	21402.03	107	N/A
	FL-BAS-2	0.46	196	21734.70	108.66	N/A
	FL-BAS-3	0.73	200	110.88	110.88	N/A
	FL-BAS-4	0.41	197	109.22	109.22	N/A
	FL-CS	0.57	185	20514.9	102.56	N/A
	FL-TOPK-BIS	0.76	200	110.88	110.88	N/A
	FL-TOP	**0.82**	200	**110.88**	**110.88**	N/A
	FL-BASIC-DP	0.59	200	22178.27	110.88	1
	FL-BAS-2-DP	0.38	200	22178.27	110.88	1
	FL-BAS-3-DP	0.56	200	110.88	110.88	1
	FL-BAS-4-DP	0.33	200	110.88	110.88	1
	FL-CS-DP	0.53	200	22178.27	110.88	1
	FL-TOPK-BIS-DP	0.68	184	102.01	102.01	0.97
	FL-TOP-DP	**0.81**	200	**110.88**	**110.88**	**1**
1%	FL-BASIC	0.71	194	21512.92	215.12	N/A
	FL-BAS-2	0.59	200	22178.27	221.77	N/A
	FL-BAS-3	0.76	200	221.77	221.77	N/A
	FL-BAS-4	0.56	195	216.23	216.23	N/A
	FL-CS	0.69	200	22178.27	221.77	N/A
	FL-TOPK-BIS	0.79	197	218.45	218.45	N/A
	FL-TOP	**0.83**	200	**221.77**	**221.77**	N/A
	FL-BASIC-DP	0.65	197	21845.59	218.45	1
	FL-BAS-2-DP	0.62	198	21956.48	219.56	1
	FL-BAS-3-DP	0.66	198	219.56	219.56	1
	FL-BAS-4-DP	0.52	198	219.56	219.56	1
	FL-CS-DP	0.66	189	20958.46	209.58	0.98
	FL-TOPK-BIS-DP	0.70	174	192.94	192.94	0.96
	FL-TOP-DP	**0.81**	183	**202.92**	**202.92**	**0.97**
5%	FL-BASIC	0.78	196	21734.70	1086.73	N/A
	FL-BAS-2	0.72	199	22067.38	1103.36	N/A
	FL-BAS-3	0.81	199	1103.36	1103.36	N/A
	FL-BAS-4	0.76	196	1086.73	1086.73	N/A
	FL-CS	0.82	200	22178.27	1108.91	N/A
	FL-TOPK-BIS	0.83	196	1086.73	1086.73	N/A
	FL-TOP	**0.84**	200	**1108.91**	**1108.91**	N/A
	FL-BASIC-DP	0.76	195	21623.81	1081.18	0.99
	FL-BAS-2-DP	0.72	195	21623.81	1081.18	0.99
	FL-BAS-3-DP	0.76	199	1103.36	1103.36	1
	FL-BAS-4-DP	0.75	191	1059.01	1059.01	0.99
	FL-CS-DP	0.78	160	17742.61	887.13	0.94
	FL-TOPK-BIS-DP	0.71	152	842.77	842.77	0.92
	FL-TOP-DP	**0.81**	152	**842.77**	**842.77**	**0.92**
10%	FL-BASIC	0.81	196	21734.70	2173.47	N/A
	FL-BAS-2	0.78	199	22067.38	2206.74	N/A
	FL-BAS-3	0.82	195	2162.38	2162.38	N/A
	FL-BAS-4	0.79	200	2217.83	2217.83	N/A
	FL-CS	0.85	182	20182.22	2018.22	N/A
	FL-TOPK-BIS	0.84	196	2173.47	2173.47	N/A
	FL-TOP	**0.85**	199	**2206.74**	**2206.74**	N/A
	FL-BASIC-DP	0.79	189	20958.46	2095.85	0.98
	FL-BAS-2-DP	0.77	189	20958.46	2095.85	0.98
	FL-BAS-3-DP	0.79	183	2029.31	2029.31	0.97
	FL-BAS-4-DP	0.78	195	2162.38	2162.38	0.99
	FL-CS-DP	0.72	167	18518.85	1851.89	0.95
	FL-TOPK-BIS-DP	0.69	138	1530.30	1530.30	0.90
	FL-TOP-DP	**0.80**	157	**1740.99**	**1740.99**	**0.93**
100%	FL-STD	0.86	200	22178.27	22178.27	N/A
	FL-STD-DP	0.56	60	6653.48	6653.48	0.76

Tab. 8.11: Summary of results on Fashion-MNIST dataset.

Compression ratio (r)	Algorithms	*Performance*					
		Bal_Acc	*AUROC*	*Round*	*Downstream Cost* (Kilobyte)	*Upstream Cost* (Kilobyte)	ϵ
0.01%	FL-BASIC	0.49	0.45	100	11948.91	1.19	N/A
	FL-BAS-2	0.49	0.45	94	11231.98	1.12	N/A
	FL-BAS-3	0.49	0.45	81	0.96	0.96	N/A
	FL-BAS-4	0.49	0.49	100	1.19	1.19	N/A
	FL-CS	-	-	-	-	-	N/A
	FL-TOP-Bis	0.59	0.63	100	1.19	1.19	N/A
	FL-TOP	**0.64**	**0.70**	60	**0.71**	**0.71**	N/A
	FL-BASIC-DP	0.49	0.45	6	716.93	0.07	0.74
	FL-BAS-2-DP	0.49	0.45	100	11948.91	1.19	1
	FL-BAS-3-DP	0.49	0.45	95	1.13	1.13	0.99
	FL-BAS-4-DP	0.49	0.47	96	1.14	1.14	0.99
	FL-CS-DP	-	-	-	-	-	-
	FL-TOP-Bis-DP	0.59	0.63	94	1.12	1.12	0.99
	FL-TOP-DP	**0.64**	**0.70**	100	**1.19**	**1.19**	**1**
0.05%	FL-BASIC	0.50	0.48	100	11948.91	5.97	N/A
	FL-BAS-2	0.49	0.46	100	11948.91	5.97	N/A
	FL-BAS-3	0.51	0.49	100	5.97	5.97	N/A
	FL-BAS-4	0.51	0.52	57	3.40	3.40	N/A
	FL-CS	0.51	0.50	100	11948.91	5.97	N/A
	FL-TOP-Bis	0.68	0.75	92	5.49	5.49	N/A
	FL-TOP	**0.68**	**0.75**	54	**3.22**	**3.22**	N/A
	FL-BASIC-DP	0.49	0.46	84	10037.08	5.02	0.96
	FL-BAS-2-DP	0.49	0.46	100	11948.91	5.97	1
	FL-BAS-3-DP	0.50	0.48	99	5.91	5.91	1
	FL-BAS-4-DP	0.52	0.51	100	5.97	5.97	1
	FL-CS-DP	0.49	0.48	100	11948.91	5.97	1
	FL-TOP-Bis-DP	0.68	0.75	92	5.49	5.49	0.98
	FL-TOP-DP	**0.68**	**0.75**	99	**5.91**	**5.91**	**1**
0.1%	FL-BASIC	0.51	0.51	99	11829.42	11.82	N/A
	FL-BAS-2	0.50	0.47	100	11948.91	11.94	N/A
	FL-BAS-3	0.53	0.53	100	11.94	11.94	N/A
	FL-BAS-4	0.50	0.53	94	11.23	11.23	N/A
	FL-CS	0.53	0.55	100	11948.91	11.94	N/A
	FL-TOP-Bis	0.69	0.76	100	11.94	11.94	N/A
	FL-TOP	**0.69**	**0.76**	68	**8.12**	**8.12**	N/A
	FL-BASIC-DP	0.50	0.49	100	11948.91	11.94	1
	FL-BAS-2-DP	0.50	0.47	100	11948.91	11.94	1
	FL-BAS-3-DP	0.55	0.56	100	11.94	11.94	1
	FL-BAS-4-DP	0.51	0.52	100	11.94	11.94	1
	FL-CS-DP	0.51	0.51	99	11829.42	11.82	1
	FL-TOP-Bis-DP	0.68	0.75	89	10.63	10.63	0.98
	FL-TOP-DP	**0.69**	**0.76**	85	**10.15**	**10.15**	**0.97**
0.5%	FL-BASIC	0.58	0.68	100	11948.91	59.74	N/A
	FL-BAS-2	0.56	0.58	99	11829.42	59.15	N/A
	FL-BAS-3	0.61	0.68	100	59.74	59.74	N/A
	FL-BAS-4	0.56	0.59	100	59.74	59.74	N/A
	FL-CS	0.66	0.71	100	11948.91	59.74	N/A
	FL-TOP-Bis	0.71	0.78	100	59.74	59.74	N/A
	FL-TOP	**0.71**	**0.79**	95	**56.76**	**56.76**	N/A
	FL-BASIC-DP	0.57	0.64	100	11948.91	59.74	1
	FL-BAS-2-DP	0.57	0.59	100	11948.91	59.74	1
	FL-BAS-3-DP	0.58	0.67	100	59.74	59.74	1
	FL-BAS-4-DP	0.54	0.57	34	20.31	20.31	0.83
	FL-CS-DP	0.61	0.68	100	11948.91	59.74	1
	FL-TOP-Bis-DP	0.68	0.75	55	32.86	32.86	0.89
	FL-TOP-DP	**0.69**	**0.76**	24	**14.34**	**14.34**	**0.80**

Tab. 8.12: Summary of results on Medical dataset (Part 1).

Compression ratio (r)	Algorithms	*Performance*					
		Bal_Acc	*AUROC*	*Round*	*Downstream Cost* (Kilobyte)	*Upstream Cost* (Kilobyte)	ϵ
1%	FL-BASIC	0.64	0.72	100	11948.91	119.49	N/A
	FL-BAS-2	0.62	0.66	100	11948.91	119.49	N/A
	FL-BAS-3	0.62	0.66	85	101.57	101.57	N/A
	FL-BAS-4	0.56	0.59	100	119.49	119.49	N/A
	FL-CS	0.68	0.75	100	11948.91	119.49	N/A
	FL-TOP-Bis	0.72	0.79	100	119.49	119.49	N/A
	FL-TOP	**0.72**	**0.79**	58	**69.30**	**69.30**	N/A
	FL-BASIC-DP	0.64	0.70	100	11948.91	119.49	1
	FL-BAS-2-DP	0.62	0.67	100	11948.91	119.49	1
	FL-BAS-3-DP	0.61	0.71	100	119.49	119.49	1
	FL-BAS-4-DP	0.57	0.66	100	119.49	119.49	1
	FL-CS-DP	0.66	0.72	100	11948.91	119.49	1
	FL-TOP-Bis-DP	0.68	0.74	53	63.33	63.33	0.89
	FL-TOP-DP	**0.69**	**0.76**	22	**26.29**	**26.29**	**0.79**
5%	FL-BASIC	0.72	0.80	100	11948.91	597.45	N/A
	FL-BAS-2	0.68	0.75	100	11948.91	597.45	N/A
	FL-BAS-3	0.69	0.76	98	585.5	585.5	N/A
	FL-BAS-4	0.66	0.72	100	597.45	597.45	N/A
	FL-CS	0.73	0.81	98	11709.93	585.5	N/A
	FL-TOP-Bis	0.72	0.79	100	597.45	597.45	N/A
	FL-TOP	**0.72**	**0.80**	95	**567.57**	**567.57**	N/A
	FL-BASIC-DP	0.69	0.76	100	11948.91	597.45	1
	FL-BAS-2-DP	0.68	0.75	98	11709.93	585.5	1
	FL-BAS-3-DP	0.65	0.71	90	537.70	537.70	0.98
	FL-BAS-4-DP	0.67	0.74	98	585.5	585.5	1
	FL-CS-DP	0.69	0.76	100	11948.91	597.45	1
	FL-TOP-Bis-DP	0.67	0.74	38	227.03	227.03	0.84
	FL-TOP-DP	**0.68**	**0.75**	23	**137.41**	**137.41**	**0.79**
10%	FL-BASIC	0.74	0.81	100	11948.91	1194.89	N/A
	FL-BAS-2	0.70	0.77	100	11948.91	1194.89	N/A
	FL-BAS-3	0.72	0.80	98	1170.99	1170.99	N/A
	FL-BAS-4	0.70	0.77	99	1182.94	1182.94	N/A
	FL-CS	0.74	0.82	100	11948.91	1194.89	N/A
	FL-TOP-Bis	0.72	0.80	100	1194.89	1194.89	N/A
	FL-TOP	**0.74**	**0.82**	90	**1075.40**	**1075.40**	N/A
	FL-BASIC-DP	0.69	0.76	99	11829.42	1182.94	1
	FL-BAS-2-DP	0.69	0.76	95	11351.46	1135.15	0.99
	FL-BAS-3-DP	0.69	0.76	95	1135.15	1135.15	0.99
	FL-BAS-4-DP	0.69	0.76	100	1194.89	1194.89	1
	FL-CS-DP	0.69	0.76	96	11470.95	1147.09	0.99
	FL-TOP-Bis-DP	0.67	0.73	37	442.11	442.11	0.84
	FL-TOP-DP	**0.68**	**0.74**	23	**274.82**	**274.82**	**0.79**
100%	FL-STD	0.74	0.82	99	11829.42	11829.42	N/A
	FL-STD-DP	0.66	0.72	62	7408.32	7408.32	0.91

Tab. 8.13: Summary of results on Medical dataset (Part 2).