Doctoral Thesis in Information and Communication Technology

Privacy preserving behaviour learning for the IoT ecosystem

SANA IMTIAZ

Stockholm, Sweden 2021

Co-funded by the
Erasmus+ Programme
of the European Union

Privacy preserving behaviour learning for the IoT ecosystem

SANA IMTIAZ

Doctoral Thesis in Information and Communication Technology
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden
and
Institute of Information and Communication Technologies,
Electronics and Applied Mathematics
Université catholique de Louvain
Louvain-la-Neuve, Belgium
December 2021
Collection Number: 851 | 2021

School of Electrical Engineering
and Computer Science
KTH Royal Institute of Technology
SE-164 40 Kista
SWEDEN

TRITA-EECS-AVL-2021:78
ISBN: 978-91-8040-067-1
UCLouvain Collection Number: 851 | 2021

Academic Dissertation which, with due permission of the KTH Royal Institute of Technology, is submitted for public defence for the Degree of Doctor of Philosophy on Friday the 17th December 2021, at 14:00 in Sal C, Electrum, Kungliga Tekniska Högskolan, Kistagången 16, Kista.

Printed by Universitetsservice US-AB

To my dear family, Chooza, and Choozi.

Abstract

IoT has enabled the creation of a multitude of personal applications and services for a better understanding and improvement of urban environments and our personal lives. These services are driven by the continuous collection and analysis of sensitive and private user data to provide personalised experiences. Among the different application areas of IoT, smart health care, in particular, necessitates the usage of privacy preservation techniques in order to guarantee protection from user privacy-breaching threats such as identification, profiling, localization and tracking, and information linkage. Traditional privacy preservation techniques such as pseudonymization are no longer sufficient to cater to the requirements of privacy preservation in the fast-growing smart health care domain due to the challenges offered by big data *volume*, *velocity* and *variety*. On the other hand, there is a number of modern privacy preservation techniques with respective overheads that may have a negative impact on application performance such as reduced accuracy, reduced data utility, and increased device resource usage. There is a need to select appropriate privacy preservation techniques (and solutions) according to the nature of data, system performance requirements and resource constraints, in order to find proper trade-offs between providing privacy preservation, data utility, and acceptable system performance in terms of accuracy, runtime, and resource consumption.

In this work, we investigate different privacy preservation solutions and measure the impact of introducing our selected privacy preservation solutions on the performance of different components of the IoT ecosystem in terms of data utility and system performance. We implement, illustrate, and evaluate the results of our proposed approaches using real-world and synthetic privacy-preserving smart health care datasets. First, we provide a detailed taxonomy and analysis of the privacy preservation techniques and solutions which may serve as a guideline for selecting appropriate techniques according to the nature of data and system requirements. Next, in order to facilitate privacy preserving data sharing, we present and implement a method for creating realistic synthetic and privacy-preserving smart health care datasets using Generative Adversarial Networks and Differential Privacy. Later, we also present and develop a solution for privacy preserving data analytics, a differential privacy library PyDPLib, with health care data as a use case.

In order to find proper trade-offs between providing necessary privacy preservation, device resource consumption and application accuracy, we present and implement a novel approach with corresponding algorithms and an end-to-end system pipeline for reconfigurable data privacy in machine learning on resource-limited computing devices. Our evaluation results show that, while providing the required level of privacy, our proposed approach allows us to achieve up to 26.21% memory, 16.67% CPU instructions, and 30.5% of network bandwidth savings as compared to making all the data private. Moreover, we also present and implement an end-to-end solution for privacy-preserving time-series forecasting of user health data streams using Federated Learning and Differential Privacy. Our proposed solution finds a proper trade-off between providing necessary privacy preservation, application accuracy and runtime, and at best introduces a decrease of $\approx 2\%$ in the prediction accuracy of the trained models.

Sammanfattning

IoT har möjliggjort skapandet av en mängd personliga applikationer och tjänster för en bättre förståelse och förbättring av stadsmiljöer och våra personliga liv. Dessa tjänster drivs av kontinuerlig insamling och analys av känslig och privat användardata för att ge personliga upplevelser. Bland de olika applikationsområdena för IoT, kräver i synnerhet smart hälsovård användningen av tekniker för bevarande av integritet för att garantera skydd mot användarnas integritetsintrång, såsom identifiering, profilering, lokalisering och spårning och informationskoppling. Traditionella tekniker för bevarande av integritet som pseudonymisering är inte längre tillräckliga för att tillgodose kraven på bevarande av integritet i den snabbväxande smarta hälsovårdsdomänen på grund av de utmaningar som stora datamängder, hastighet och variation forcerar. Å andra sidan finns det ett antal moderna tekniker för bevarande av integritet med respektive omkostnader som kan ha en negativ inverkan på applikationsprestanda såsom minskad noggrannhet, minskad datanytta och ökad resursanvändning på enheten. Det finns ett behov av att välja lämpliga sekretessskyddstekniker (och lösningar) i enlighet med datas natur, systemprestandakrav och resursbegränsningar, för att hitta korrekta avvägningar mellan tillhandahållande av integritetsbevarande, dataverktyg och acceptabel systemprestanda i form av av noggrannhet, körtid och resursförbrukning.

I detta arbete undersöker vi olika lösningar för bevarande av integritet och mäter effekten av att introducera våra utvalda lösningar för bevarande av integritet på prestandan hos olika komponenter i IoT-ekosystemet när det gäller datanytta och systemprestanda. Vi implementerar, illustrerar och utvärderar resultaten av våra föreslagna tillvägagångssätt med hjälp av verkliga och syntetiska integritetsbevarande smarta hälsodatauppsättningar. Först tillhandahåller vi en detaljerad taxonomi och analys av tekniker och lösningar för bevarande av integritet som kan fungera som en riktlinje för att välja lämpliga tekniker i enlighet med typen av data och systemkrav. Därefter, för att underlätta integritetsbevarande datadelning, presenterar och implementerar vi en metod för att skapa realistiska syntetiska och integritetsbevarande smarta hälsovårdsdatauppsättningar med hjälp av Generative Adversarial Networks och Differential Privacy. Senare presenterar och utvecklar vi också en lösning för integritetsbevarande dataanalys, ett differentiellt integritetsbibliotek PyDPLib, med sjukvårdsdata som ett användningsfall.

För att hitta korrekta avvägningar mellan tillhandahållande av nödvändig integritetsbevarande, enhetsresursförbrukning och applikationsnoggrannhet presenterar och implementerar vi ett nytt tillvägagångssätt med motsvarande algoritmer och en end-to-end systempipeline för omkonfigurerbar datasekretess i maskininlärning på resursbegränsade datorenheter. Våra utvärderingsresultat visar att, samtidigt som vi tillhandahåller den nödvändiga integritetsnivån, tillåter vårt föreslagna tillvägagångssätt oss att uppnå upp till 26,21% minne, 16,67% CPU-instruktioner och 30,5% av besparingar på nätverkets bandbredd jämfört med att göra all data privat. Dessutom presenterar och implementerar vi också en helhetslösning för integritetsbevarande tidsserieprognoser för användarhälsodataströmmar med hjälp av Federated Learning och Differential Privacy. Vår föreslagna lösning finner en lämplig avvägning mellan att tillhandahålla nödvändig integritetsbevarande, applikationsnoggrannhet och körtid, och introducerar i bästa fall en minskning med $\approx 2\%$ i prediktionsnoggrannheten för de tränade modellerna.

Résumé

L'IoT a permis la création d'une multitude d'applications et de services personnels pour une meilleure compréhension et amélioration des environnements urbains et de nos vies personnelles. Ces services sont alimentés par la collecte et l'analyse continues de données sensibles et privées des utilisateurs afin de fournir des expériences personnalisées. Parmi les différents domaines d'application de l'IoT, les soins de santé intelligents, en particulier, nécessitent l'utilisation de techniques de préservation de la vie privée afin de garantir la protection contre les menaces d'atteinte à la vie privée des utilisateurs, telles que l'identification, le profilage, la localisation et le suivi, et la mise en relation des informations. Les techniques traditionnelles de préservation de la vie privée, telles que la pseudonymisation, ne suffisent plus à répondre aux exigences de préservation de la vie privée dans le domaine en plein essor des soins de santé intelligents, en raison des défis posés par le *volume*, la *vitesse* et la *variété* des données. D'autre part, il existe un nombre de techniques modernes de préservation de la confidentialité avec des frais généraux respectifs qui peuvent avoir un impact négatif sur les performances de l'application, comme une précision réduite, une utilité réduite des données et une utilisation accrue des ressources du système. Il est nécessaire de sélectionner des techniques (et des solutions) appropriées de préservation de la confidentialité en fonction de la nature des données, des exigences de performance du système et des contraintes de ressources, afin de trouver des compromis appropriés entre la préservation de la confidentialité, l'utilité des données et une performance acceptable du système en termes de précision, de temps d'exécution et de consommation de ressources.

Dans ce travail, nous étudions différentes solutions de préservation de la vie privée et mesurons l'impact de l'introduction de nos solutions de préservation de la vie privée sélectionnées sur les performances de différents composants de l'écosystème IoT en termes d'utilité des données et de performances du système. Nous mettons en œuvre, illustrons et évaluons les résultats de nos approches proposées en utilisant des ensembles de données de soins de santé intelligents réels et synthétiques préservant la confidentialité. Tout d'abord, nous fournissons une taxonomie et une analyse détaillées des techniques et solutions de préservation de la vie privée qui peuvent servir de ligne directrice pour sélectionner les techniques appropriées en fonction de la nature des données et des exigences du système. Ensuite, afin de faciliter le partage de données préservant la vie privée, nous présentons et mettons en œuvre une méthode pour créer des ensembles de données synthétiques réalistes et préservant la vie privée dans le domaine des soins de santé intelligents en utilisant des réseaux adversariaux génératifs et la confidentialité différentielle. Par la suite, nous présentons et développons également une solution pour l'analyse de données préservant la confidentialité, une bibliothèque de confidentialité différentielle PyDPLib, avec des données de soins de santé comme cas d'utilisation.

Afin de trouver des compromis appropriés entre la préservation nécessaire de la confidentialité, la consommation de ressources du dispositif et la précision de l'application, nous présentons et mettons en œuvre une nouvelle approche avec les algorithmes correspondants et un pipeline système de bout en bout pour la confidentialité des données reconfigurable dans l'apprentissage automatique sur des appareils à ressources limitées. Nos résultats d'évaluation montrent que, tout en assurant le niveau de confidentialité requis, l'approche proposée permet d'économiser jusqu'à 26,21% de mémoire, 16,67% d'instructions CPU et 30,5% de bande passante réseau par rapport à la confidentialité de toutes les données. En outre, nous présentons et mettons en œuvre une solution de bout en bout pour la prévision de séries temporelles préservant la confidentialité des flux de données de santé en utilisant l'apprentissage fédéré et la confidentialité différentielle. La solution que nous proposons trouve un compromis approprié entre la préservation de la confidentialité, la précision de l'application et le temps d'exécution, et introduit au mieux une diminution de $\approx 2\%$ de la précision de prédiction des modèles formés.

Acknowledgements

The acknowledgements Section is perhaps the most difficult to write in a doctoral thesis, simply because a doctoral thesis is the fruit of collective efforts from a large group of people (supervisors, mentors, colleagues, administrative staff, friends and family) over a long period of typically 4-6 years. All this hard work and support can not possibly be summed up in a small chapter but here goes a small effort to thank all the wonderful people who have been a part of my doctoral journey.

My foremost gratitude goes to my primary supervisors, Vladimir Vlassov (Vlad) and Ramin Sadre. In my culture, a teacher is considered to be a spiritual father. And professors like Vlad go the extra mile to not only be an academic advisor but also be a good adviser at life for his students. Vlad created a very healthy atmosphere in our research group that allowed us to become not only colleagues who support each other's research but also be an academic family that thoroughly enjoys their group meetings! At the end of 3 years of my stay at KTH, I can say that Vlad has become my academic dad who is always available to support my work in both technical and administrative capacities, and is always there to give me the right amount of push to keep myself motivated at work and to meet my deadlines in time. Your ideas and recommendations have shaped and improved the most of my research work. I thank you for the super useful brainstorming and problem-solving sessions, and for encouraging me to collaborate with colleagues. I also thank you, Vlad, for believing in me at times when I myself could not, particularly when I hit hurdles during my research. I will miss your energy, your valuable ideas and insights into research, your unique collection of proverbs, and our lively group meetings.

My primary supervisor at UCLouvain, Ramin Sadre, thank you for your very valuable ideas, guidance, patience and cooperation throughout my doctoral journey. Your support at the beginning of my doctoral research was vital to my whole academic journey. Your ideas and recommendations have always shaped my research direction and improved the presentation of my research work. Thank you for always being there for the quick reviews and editing right at the submission deadlines, most importantly, for always helping me with the French translations.

My co-supervisor at KTH, Sarunas Girdzijauskas, you have always been very supportive and available whenever I needed help with the research ideas or even computing resources for master thesis students. I thank you for the brainstorming sessions and your valuable ideas and feedback regarding my research. My doctoral support committee, Peter Van Roy (UCLouvain) and Sonja Buchegger (KTH), I am truly grateful for your support and very valuable ideas for shaping my research. Peter, thank you for your valuable suggestions for improving my work, for asking the right questions that helped me shape my research pitch, and for always being a source of positive energy and support. Sonja, thank you for welcoming me to your weekly group meetings on privacy that gave me the perfect exposure to a wide spectrum of privacy preservation solutions I that ended up applying in my work. I also thank you for always being available for analyzing the privacy aspects of my research works, for asking the right questions that make me think and analyze my work better, for providing useful suggestions to improve my experiments, evaluation and presentation; and for the positive energy you impart in every interaction.

My internal thesis reviewer at KTH, Elena Troubitsyna, thank you for your timely and detailed comments that helped in improving my thesis. Anne Håkansson, my evaluator for 80% doctoral seminar at KTH, thank you for your in-depth feedback and very valuable suggestions for improving the presentation of my thesis. Thank you to all the committee members at my private doctoral defence, Yao Guo (Peking University, China), Monowar Bhuyan (Umeå University, Sweden), Siegfried Nijssen and Charles Pecheur (UCLouvain), and Sonja Buchegger. Your in-depth feedback and valuable suggestions have greatly improved my thesis.

I would also like to thank my collaborators and co-authors Zainab Abbas, Muhammad Arsalan, Hassan Nazeer Chaudhry, Philipp Matthies, Francisco Pinto, and Frida Schain, it was a pleasant and great learning experience working with you all. Special thanks to my talented master thesis students Sonia-Florina Horchidan, Zannatun Tania and Stefano Fedeli for their hard work and collaboration. Thank you for contributing to my work!

I would also like to thank the administrative teams at KTH and UCLouvain for their support. Thank you to our head of department, Thomas Sjöland, for his kindness and immense support with my doctoral career at KTH, for always providing a positive outlook on research, and for encouraging us to participate in research events during my stay at KTH. Thomas, your open door policy makes all of us at the department feel welcome and cared for. I will miss our coffee breaks and in-depth discussions on academia, administration, and life in general. I would like to thank Christian Schulte for helping me integrate smoothly when I first arrived at KTH. Thank you for introducing me to so many good colleagues from other divisions that I ended up having a great time with during my stay at KTH. I also thank you for always being my go-to translator for Swedish for academic matters and everyday things like insurance matters for my rabbits. Your kindness and positive energy will never never be forgotten. Special thanks to Susy Matthew and Madeleine Printzsköld at KTH for handling all my administrative matters swiftly and smoothly. Special thanks to Vanessa Maons and Sophie Renard for their magically swift and efficient support on the administrative matters at UCLouvain, and for the positive energy that you ladies radiate. I thank all you ladies for facilitating the administrative parts of my doctoral journey.

Special token of thanks to a lovely friend and the coolest office mate ever, Zainab Abbas. Your singing made the otherwise gloomy winters in Stockholm very enjoyable :) Thank you for your wisdom and kindness, for the productive brainstorming sessions supported by tea breaks, your ideas and collaboration at research, for co-supervising the master theses, for your immense support throughout my doctoral thesis, for making my office a happy place to work at, for providing me 24/7 logistics support (Thank you, Mudassir!), and for all the memorable trips we did together during our doctoral thesis. You have simply been the best part of my stay at KTH! Another special token of thanks to my dear friend and collaborator, Muhammad Arsalan. Thank you for your valuable ideas and contributions to my research, for your active collaboration, the hectic yet productive pair-programming sessions, and the near-submission deadline editing sessions on Overleaf. I thank you for your kindness, patience, and immense support through my doctoral journey, and mostly, for your sense of humour that always lightened up our meetings with master thesis students.

My research group members, Sina, Tianze, Klas, David, and Desta, thank you for the fruitful discussions in group meetings and for the wonderful memories at KTH. Special thanks to my EMJD-DC friends Bilal, Igor, Khulan, and Jawad, for inspiring me and encouraging me during my work, and for their research ideas and support during the initial years of my doctoral journey. To my mentors Amira Soliman, Shatha Jaradat, and Leila Bahri, thank you for inspiring and mentoring me during my doctoral studies. My dear friends Shazem, Nancy, Freya, Mehar ul Nisa, Hadi, Farah, Asra, Hira, Hira, Anum, Anum (yes there are two Hira's and Anum's), Sanna, Saleha, and Mona, thank you for your support and encouragement. To my teachers from Pakistan, Saira Aziz, Abdul Wadood, and Irfan Ahmed, I owe it all to the exceptional teachers like you. Thank you for your wisdom, for your belief in me and the never-ending support through the 12+ years of my academic life.

My dear parents, thank you ammi and abbu for your constant love, prayers, belief, patience, and support. I surely would not have come this far without you. I am grateful to have you as my parents, for you have always provided a very happy, loving and healthy environment at home. Thank you for the never-ending hard work you put in for raising your daughters. You have challenged the norms of our society by raising us siblings as strong and educated women; and have always given us all the best, be it at education or quality of life. My lovely sisters Sahar, Hira and Tooba, you are my constant source of love, laughter, inspiration and hope. Sahar, thank you for being my pillar of support during my stay in Sweden, for the yummy food, and for taking care of my bunnies. My wonderful sisters, thank you for your support in my academic career, your patience during my all-nighters throughout my academic life, for the laughter and entertainment you provide on a daily basis, for your love and support, and for always being there for me. My bunnies, Chooza and Choozi, thank you for your constant companionship and love during my academic journey in Europe. I wish you could have lived longer...

Last but not the least, this work was supported by the Erasmus Mundus Joint Doctorate in Distributed Computing (EMJD-DC) funded by the Education, Audiovisual and Culture Executive Agency (EACEA) of the European Commission under the FPA 2012-0030, and FoFu at KTH.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER

1

Introduction

1.1 Preface

Current trends in technology show that the use of IoT devices for improving quality of life is on the rise in a multitude of application domains such as smart buildings and living, smart healthcare, smart environment, smart city, smart energy, smart transport and mobility, smart manufacturing and retail, and smart agriculture. However, the data being generated by IoT devices is generally of a private and sensitive nature, especially when it comes to health care related applications. Data is continuously being acquired by a variety of IoT sensors, and this data is transformed through some edge devices before processing over cloud platforms, which leads to data integrity and privacy concerns. Privacy preservation is needed in the data collection, aggregation, storage and retrieval processes. Traditional privacy preservation techniques such as pseudonymization are not able to cater to the challenge of *velocity*, *variety* and *volume* that comes with big data. Therefore, novel techniques and solutions are required to address the problem of user privacy preservation. Although each IoT domain possesses unique challenges in ensuring user privacy, smart health care is one of the most challenging IoT domains as it deals with the collection and processing of highly sensitive and private data. In this thesis, we aim to improve user privacy preservation by applying privacy preservation solutions to different components of the IoT ecosystem, within the context of the smart health care domain.

The first step towards privacy preservation in the IoT ecosystem is to develop an understanding of the different components of the IoT ecosystem.

IoT Ecosystem. Internet of Things is a term coined originally by Kevin Ashton in 1999 [1]. It denotes the set of interconnected sensors and devices which are fundamental units for collecting data that drives modern day applications and services. Broadly speaking, any sensor that is capable of collecting data, processing it using built-in circuitry and transmitting it qualifies as a smart sensor. This interconnected set of sensors, coupled with the ability to connect to the internet, collectively form the Internet of Things. The term IoT ecosystem encompasses all the software and hardware components that collectively provide big data-driven services. Mazhelis et al. [2] observe that "Since the essence of the IoT is the interconnection of the physical world of things with the virtual world of Internet, the software and hardware platforms, as well as the standards commonly used for enabling such interconnection, may become a core of an IoT ecosystem". In the context of this thesis, we focus on the following components of the IoT ecosystem: data, resource-constrained edge devices, ML applications, and data analytics and visualization.

Privacy Preservation. When it comes to privacy preservation techniques, a wide variety of solutions is available. Typically, all privacy preservation techniques employ mechanisms like information flow control [3], data or model obfuscation via noise addition [4], use of cryptography [5, 6], anonymization via generalization and suppression of attributes [7–9], and use of private computation units [10]. As observed in [11], all these techniques have their respective performance overheads such as increased resource usage, reduced model accuracy and longer processing times. Therefore, lightweight, scalable and efficient privacy preservation techniques should be applied to the components of the IoT ecosystem that can cater to the requirements of volume, velocity and variety. Doing so will allow finding proper trade-offs between preserving necessary user privacy, retaining data utility, acceptable application performance and model accuracy, and low overhead on device resource usage.

Smart Health Care. Smart health care is a fast-growing IoT domain with an impact on two user groups: individuals and society. Individuals focus on the ability to track physical activities and diet patterns by means of wearable trackers and diet logging applications, whereas the societal goal is to increase the quality of medical treatment while reducing healthcare costs, and provide better and personalized health care [12]. The smart health care industry relies on the availability of large-scale health datasets in order to benefit from machine learning-based services. However, this health data should be collected and processed according to local privacy laws like EU's General Data Protection Regulation (GDPR) [13]. Therefore, the smart health care domain possesses additional challenges in user privacy provision: 1) there is a lack of freely available data to experiment with, as the data is of a highly sensitive and private nature, 2) there are limitations on sharing data with protected health information due to the threat of misuse or re-identification, 3) smart health care data possesses highly diverse data types and formats, and bounded ranges, 4) health care data requires high utility and offers low tolerance towards data perturbance, and 5) there is low tolerance towards loss of accuracy on the developed applications and services due to the sensitive nature of health care data. All these unique challenges make smart health care a particularly challenging IoT domain for ensuring privacy preservation.

Resource-constrained Edge Devices. Edge devices are vital components of the IoT ecosystem that are responsible for collecting and aggregating the data from sensor nodes before sending it to the cloud platforms for processing. Traditionally, an edge device is an embedded system with a small microcontroller unit (MCU) for processing. The MCUs in edge devices are resource constrained due to their limited memory footprint, fewer computation cores, and low clock speeds. Moreover, these devices often have limited battery power and access to physical memory. These limitations constrain the deployment and execution of complex privacy preservation techniques and also, the machine learning models on MCUs [14]. As we stated earlier, privacy preservation techniques require increased resource usage. Therefore, there is a need to find appropriate trade-offs between necessary privacy provision and increased device resource usage on these resource-constrained edge devices.

Streaming Data and Stream Processing. Streaming data is the continuous flow of data generated by various sources. The data collected by IoT devices often comes in the form of data streams which are aggregated and processed in batches or as real-time data streams. The past decades have witnessed a dramatic increase in the real-time data processing needs which has led to the popularity of high performance distributed stream processing systems, such as Apache Flink [15], Apache Storm [16], and Spark Streaming [17]. In terms of privacy preservation on data streams and stream processing systems, there is a rather narrow spectrum of privacy preserving techniques that offer minimal negative impact on system efficiency. Particularly, in the context of smart health care data streams, there is a two-fold constraint of low loss of accuracy and system efficiency apart from providing necessary user privacy preservation.

Private Data Analytics. As observed by Wieringa et al. [18], the conventional wisdom assumes that the goals of data analytics and privacy protection are contradictory. However, privacy preserving techniques such as anonymization, perturbation and obfuscation can enable privacy-preserving data analytics. Common privacy threats in data analytics include re-identification, attribute and information disclosure, profiling, linkage and data inference. Since the privacy preserving techniques tend to adversely affect data utility, there is a need to find a proper trade-off between data utility and risk of information disclosure for private data analytics systems.

1.2 Thesis Statement

Sharing and processing data with sensitive information raises privacy concerns and hence, requires privacy preservation. Finding a proper trade-off between preserving necessary user privacy and achieving (best-effort) acceptable system performance by means of applying (combinations of) appropriate privacy preservation techniques, allows to reduce (if not avoid) the potential negative impact of applying privacy preservation techniques on the Machine Learning-based systems performance including reduced model accuracy, slower runtimes, and increased device resource usage.

1.3 Objectives and Research Questions

Provision of user privacy preservation is inevitable for all the emerging domains in the IoT ecosystem. Traditional privacy preservation mechanisms such as removing identifiable information are no longer sufficient for privacy preservation due to the *volume*, *variety* and *velocity* of the collected data. Moreover, each privacy preservation technique comes with its limitations and impact on the data utility, application accuracy and system efficiency. Therefore, it is important to not only systematically classify the privacy preservation techniques applicable to the IoT ecosystem, but also analyze the impact of introducing privacy preservation techniques on the system performance. Moreover, there is a need to design efficient solutions for privacy preserving data sharing in order to minimize the impact of privacy breaching attacks and to encourage the development and improvement of big-data driven ML-based services. In this thesis, we focus on privacy preservation with the smart health care domain as a use case as it is a fast-growing domain possessing its own unique and complex challenges, as presented in 1.1 (Smart Health Care).

This thesis has the following objectives with the associated corresponding research questions. Our first objective is to taxonomize and analyze the privacy preservation techniques applicable to the IoT ecosystem (O1) and the second objective is to design solutions for privacy preserving data sharing and analytics (O2). The third objective is to find a proper trade-off between privacy preservation, data utility and acceptable system performance (O3), and lastly, to evaluate the impact of introducing privacy preserving solutions on the performance of different components of the IoT ecosystem (O4).

O1 **Analysis of the state of the art and taxonomy of privacy preserving techniques for the IoT ecosystem.** There is a multitude of privacy preservation techniques that may not all be applicable to the unique nature of big data and corresponding processing systems. There is a need to develop a detailed taxonomy of privacy preservation techniques (and respective solutions) which may serve as a guideline to select appropriate solutions for privacy preservation according to their strengths and limitations. There is also a need to analyze the potential trade-offs of using these privacy preservation solutions such as the provision of necessary user privacy, achieving acceptable application accuracy and system efficiency, so that one may make the best choice according to the system design requirements. We describe more about the systematic classification of privacy preservation techniques in Chapter 3 and Paper [11].

Research Question. How to select appropriate privacy preservation techniques and solutions for different components of the IoT ecosystem?

O2 **Design and evaluation of effective and efficient solutions for privacy-preserving data sharing and analytics.** Data is a key component of all IoT-based applications and services. In the context of smart health care, the nature of collected data is extremely sensitive as the *volume* of this data may create unique user profiles. Therefore, traditional solutions for privacy preservation such as pseudonymization are not sufficient to guarantee user privacy. On the other hand, the smart health care industry relies on the availability of these large scale datasets for the provision of useful applications and services. There is a need to investigate and evaluate newer and more complex privacy preservation techniques in order to design efficient solutions for privacy-preserving data sharing and to allow third parties to perform privacy-preserving data analytics without breaching user privacy. Details on how we designed these solutions are presented in Chapters 4 and 7 and Papers [19] and [20].

Research Question. How to share useful data and analytics in an efficient and privacy-preserving manner?

O3 **Finding a proper trade-off between user privacy, data utility, acceptable system performance and increased device resource usage.** The IoT-driven applications and services require user privacy preservation but some privacy preserving techniques may negatively impact data utility or the system performance goals. Moreover, ensuring user privacy while learning from the acquired IoT sensor data, using limited available compute resources on edge devices, is a challenging task. Ideally, it is desirable to make all the features of the collected data private but due to the system performance goals (efficiency and accuracy), resource limitations or the potential negative impact on data utility, it is not always possible as doing so may affect the performance of the whole system. Therefore, there is a need to find a proper trade-off between preserving a necessary level of user privacy, data utility and achieving acceptable system performance in IoT-driven ML systems, and on resource-constrained edge devices. Chapters 5, 6 and 7, and Papers [20–22] describe how we fulfill this objective in detail.

Research Questions. How to find a proper trade-off between preserving necessary user privacy and achieving acceptable system performance? How to find this trade-off in the case of resource-constrained devices?

O4 **Evaluating the impact of applying privacy preservation solutions on the performance of different components of the IoT ecosystem.** Once we find a proper trade-off between preserving necessary user privacy, data utility and achieving acceptable system performance, we can measure the impact of introducing the privacy preservation solutions on the performance of different components of the IoT ecosystem. Factors to be considered are: retained data utility (how far off are the noisy or perturbed data distributions from the original distributions), impact on device resource usage and application accuracy, and the impact on system efficiency (in terms of application runtime). Chapters 4, 5 and 6 and papers [19,21,22] describe the measured impact of introducing the privacy preservation solutions on the data utility and system performance in detail.

Research Question. How do privacy preservation solutions impact the performance of the IoT-driven systems? Apart from the potential negative impacts, are there any benefits on the system performance (in terms of application accuracy or runtime)?

1.4 Contributions

We perform an empirical study of the impact of introducing privacy preservation techniques on the system performance and contribute to the different aspects of the IoT ecosystem with the smart health care domain as a use case. The main contributions of this dissertation addressing the aforementioned objectives are as follows.

C1 *The IoT ecosystem.* **An analysis and taxonomy of privacy preserving techniques and solutions.** We survey the state of the art of privacy preservation techniques that are applicable to the IoT ecosystem. We also develop a taxonomy of these techniques and respectively designed solutions for privacy preservation, and analyze the IoT ecosystem levels at which privacy is addressed by each solution as well as their robustness to privacy breaching attacks as part of our objective *O1*. An analysis of functional and non-functional limitations of each solution is presented, and the impact of the application of each solution to the IoT ecosystem is analyzed in terms of the offered performance trade-offs. We also identify open issues in the privacy preserving solutions when used in IoT environments.

C2 *Data.* **A novel method for synthetic private data generation.** Representative synthetic datasets are a promising solution for privacy preserving data sharing. We design and implement a novel method for synthetic and private data generation to generate realistic non-sensitive datasets from the sensitive datasets, in accordance with our objective *O2*. Generative adversarial networks (GANs) [23] have emerged as the most impressive generative models for synthetic data generation. We propose a GAN model coupled with differential privacy mechanisms for generating a realistic and private smart health care dataset. Our proposed approach is not only able to generate realistic synthetic data samples but also the differentially private data samples under different settings: learning from a noisy distribution or noising the learned distribution. We tested and evaluated our proposed approach using a real-world Fitbit dataset. Our results indicate that our proposed approach is able to generate a quality synthetic and differentially private dataset that is enriched and also preserves the statistical properties of the original dataset (objectives *O2* and *O4*).

C3 *Resource-constrained edge devices.* **Algorithms and end-to-end system pipeline for reconfigurable privacy preservation.** Ensuring user privacy on the IoT sensor data using resource-limited edge devices is a challenging task. Regardless of the resource availability, some data features must be essentially private. All other data features that may pose low privacy threats are termed non-essential features. We use the generalization techniques for data anonymization and provide customized injective privacy encoder functions to make data features private. We propose *Dynamic Iterative Greedy Search (DIGS)*, a novel approach with corresponding algorithms to select the set of optimal data features to be private for machine learning applications provided device resource constraints. DIGS selects the necessary and the most private version of data for the application, where all essential and a subset of non-essential features are made private on the edge device without resource overutilization. Our evaluation results show that, while providing the required level of privacy, DIGS allows to achieve up to 26.21% memory, 16.67% CPU instructions, and 30.5% of network bandwidth savings as compared to making all the data private (objective *O3*). Moreover, our chosen privacy encoding method has a positive impact on the accuracy of the classification model for our chosen application (objective *O4*).

C4 *ML application.* **An end-to-end pipeline for privacy preserving forecasting of data streams.** Privacy preservation plays a vital role in health care applications as the requirements for privacy preservation are very strict in this domain. Federated learning (FL) facilitates private training of ML models as it does not directly access the raw data. However, FL alone does not guarantee sufficient privacy. Differential privacy (DP) is a well-known solution for privacy preservation but DP needs to make a trade-off between privacy and accuracy due to noise addition. We design and implement an end-to-end pipeline using DP and FL for the first time in the context of health data streams. We propose a clustering mechanism to leverage the similarities between users to improve the prediction accuracy as well as significantly reduce the model training time. Depending on the dataset and features, our predictions are no more than 0.025% far off the ground-truth value with respect to the range of values. Moreover, our clustering mechanism brings a significant reduction in the training time, with up to 49% reduction in prediction accuracy error in the best case, as compared to training a single model on the entire dataset. Our proposed privacy preserving mechanism at best introduces a decrease of ≈2% in the prediction accuracy of the trained models (objective *O4*). Furthermore, our proposed clustering mechanism reduces the prediction error even in highly noisy settings by as much as 38% as compared to using a single federated private model (objective *O3*).

C5 *Data analytics and visualization.* **A library for private data analytics and visualization.** Pharmaceutical and medical technology companies accessing real-world medical data are not interested in personally identifiable data but rather in statistical aggregates, patterns, and trends. We present *PyDPLib*, a Python Differential Privacy library for private medical data analytics to facilitate the medical institutions to share data for analytics in accordance with objective *O2*. We illustrate the results of using PyDPLib for visualizing private statistics on a database of prostate cancer patients. Our experimental results show that PyDPLib allows creating statistical data plots without compromising patients' privacy while preserving underlying data distributions (objective *O4*). Moreover, PyDPLib is designed to be general enough for providing differential privacy on data in any data analytics and visualization platform, service or application.

1.5 Research Methodology

This Section presents an overview of the research methods used in our study. We first describe our general approach, followed by our implementation choices, and our methods for experimental evaluation. Finally, we present a brief account of the challenges we faced during our research.

1.5.1 General Approach

This work performs exploratory research for privacy preservation in the smart health care IoT domain in different settings: data sharing without information disclosure, privacy preservation with limited device resources, privacy preservation in near real-time applications, and privacy preservation in data analytics. We first performed a literature survey and developed a taxonomy of privacy preservation techniques and analyzed them qualitatively for strengths, weaknesses, functional and non-functional limitations, and potential trade-offs they offer (Chapter 3). Afterwards, we ventured into the acquisition of a smart health care dataset for testing our hypotheses regarding the impacts of privacy preservation techniques on the performance of different components in the IoT ecosystem. We collected a smart health care dataset using Fitbit [24] devices with the help of 25 volunteers (who gave their full consent for usage) distributed geographically around Belgium and Sweden. This data collection process took almost a year. We started working on our streaming application in parallel and developed an end-to-end pipeline for time-series forecasting health care data streams. Afterwards, we designed and implemented a GAN-based method for generating synthetic and private smart health care datasets and used it to enrich, augment and synthesize a large dataset based on our collected Fitbit dataset (Chapter 4). We performed experiments to quantify the impact of introducing privacy preservation techniques on the performance of the system for time-series forecasting in terms of accuracy and application runtime (Chapter 6). We then considered the case of resource-constrained edge devices and how the privacy preservation techniques would perform in those settings. Hence, we developed a set of algorithms to enable reconfigurable privacy preservation given the device resource constraints (Chapter 5). We also performed an industrial internship for private analytics on medical data and developed *PyDPLib*, a differential privacy library for private medical data analytics, and tested it on medical data collected across hospitals and clinics in Germany (Chapter 7). All these contributions are published as publications P1–P5.

1.5.2 Implementations

All the implementations in this research are done in Python. The volunteers provided us with data collected by Fitbit devices using the Fitbit platform. We used Fitbit API to understand and aggregate the different data fields, and used translation software to convert some of the logged data to English. Afterwards, we used Nutritionix API to impute the missing values for a caloric breakdown of meals, and our Python scripts to fix other inconsistencies in data. Keras was used to implement the smart health care data generation method using GANs. Afterwards, Tensorflow and Apache Flink were used to implement the pipeline for time-series forecasting of health data streams. The algorithms and supporting pipeline for reconfigurable privacy preservation was designed using a combination of Python libraries (NumPy, SciPy, scikit-learn, guppy and more). Google Colab, Jupyter notebook and PyCharm were used as the development environments. The remaining implementations as part of the industrial internship are confidential apart from the usage of Python libraries.

1.5.3 Experimental Evaluation

We used the latest stable versions of the aforementioned open-source platforms. Our major dataset sources were volunteers distributed across Belgium and Sweden, for the work done in Chapters 4–6. Moreover, we used a freely available dataset with data collected from MyFitnessPal platform [25, 26] for part of our work in Chapter 6. The work in Chapter 7 was done during a research internship at Health Data Pioneers (now a part of Smart Reporting GmbH) and the dataset provided was anonymised for privacy preservation and kept for private use only by the organisation. The machines used throughout the course of our research consist of both on-premises devices and virtualised environments. In both cases, we installed the necessary software and carefully performed all the experiments to avoid interference from other applications running in parallel.

1.5.4 Challenges

During the first two years of our research, we made extensive efforts for acquiring smart health care datasets that contained private information. We experienced huge public interest in knowing the results of our research during our interactions and presentations in industrial events, conferences and summer schools, but a lack of willingness to contribute towards data sharing and availability. We also contacted Fitbit in an effort to acquire data for collaborative research, but they could only offer a discount on the mass purchase of (> 50) devices. Obtaining fully informed consent of participating individuals in data collection was a major concern. Hence, we purchased 12 Fitbit devices and found volunteers around the campus and through personal/professional connections who agreed to willingly share their data for research. The participant pool consisted of users with varying levels of motivation and we had to continuously keep them engaged and motivated for the data collection process. Some of the participants used different languages for logging data, which required additional effort in translating particularly the meals to nearest-matching variant in English with correct proportions in order to record accurate nutritional activity for all the users. Since the smart health care domain is rather new in research for privacy preservation, it was difficult to find related work for some of our works. There is plenty of research available on the privacy preservation of electronic health records, but there is a huge room for research in privacy preservation in the smart health care domain. We are truly thankful to the volunteers who provided the data for this study, and to our peers and collaborators for always keeping us informed of the recent research in the smart health care domain.

1.6 Publications

The results presented in this thesis are published in conference and workshop papers as following:

P1 **On the Case of Privacy in the IoT Ecosystem: A Survey**, Sana Imtiaz, Ramin Sadre, and Vladimir Vlassov. 2019 International Conference on Internet of Things (iThings), IEEE, Atlanta, Georgia, USA, July 14-17, 2019. [11]

Contribution The author of this dissertation brainstormed, developed a taxonomy of privacy preservation techniques and analyzed corresponding privacy preservation solutions, identified the merits and limitations of each technique/approach, and wrote the entire paper.

P2 **Privacy Preserving Time-Series Forecasting of User Health Data Streams**, Sana Imtiaz, Sonia-Florina Horchidan, Zainab Abbas, Muhammad Arsalan, Hassan Nazeer Chaudhry and Vladimir Vlassov. 2020 IEEE International Conference on Big Data (Big Data), Atlanta, Georgia, USA [virtual], December 10-13, 2020. [21]

Contribution The author of this dissertation brainstormed, helped in design and development of the forecasting pipeline, provided the datasets (2/3) for experimentation, identified the appropriate privacy preservation techniques to apply on the pipeline as well as their parametric settings, analyzed the results of the experiments, and majorly participated in the paper writing.

P3 **Synthetic and Private Smart Health Care Data Generation using GANs**, Sana Imtiaz, Muhammad Arsalan, Vladimir Vlassov and Ramin Sadre. 2021 International Conference on Computer Communications and Networks (IC-CCN), IEEE, Athens, Greece [virtual], July 19-22, 2021. [19]

Contribution The author of this dissertation collected, cleaned and organized the real-world dataset for this work. The author also brainstormed, helped with design and development of the GAN pipeline, performed the experiments, and wrote the majority of text in the paper.

P4 **PyDPLib: Python Differential Privacy Library for Private Medical Data Analytics**, Sana Imtiaz, Philipp Matthies, Francisco Pinto, Máté Maros, Holger Wenz, Ramin Sadre and Vladimir Vlassov. IEEE International Conference on Digital Health (ICDH), IEEE, Chicago, USA [virtual], September 5-11, 2021. [20]

Contribution The author of this dissertation brainstormed, designed and developed PyDPLib with all its functionalities, performed the experiments, analyzed the results and wrote all the paper except the Section on Structured Clinical Data Collection.

P5 **Machine Learning with Reconfigurable Privacy on Resource-Limited Computing Devices**, Sana Imtiaz, Zannatun Tania, Hassan Nazeer Chaudhry, Muhammad Arsalan, Ramin Sadre and Vladimir Vlassov. 14th IEEE International Conference on Security, Privacy, and Anonymity in Computation, Communication, and Storage (IEEE SpaCCS 2021), New York, USA [virtual], September 30-October 3, 2021 (to appear). [22]

Contribution The author of this dissertation brainstormed, helped design and develop the algorithms and supporting pipeline, provided datasets for experimentation, analyzed the results, and wrote the majority of the paper.

1.6.1 Other Papers

Other works done during the doctoral studies but not included in this thesis are as follows.

1. **Privacy Preserving Survival Prediction**, Stefano Fedeli, Frida Schain, Sana Imtiaz, Zainab Abbas and Vladimir Vlassov. 2021 IEEE International Conference on Big Data (IEEE BigData), Orlando, Florida, USA [virtual], December 14-17, 2021 (to appear). [27]

1.7 Thesis outline

The rest of this thesis is organized as follows. Chapter 2 provides background on privacy preservation by design and default, generative adversarial networks, differential privacy, anonymization, federated learning and time-series forecasting. In Chapter 3 we present a taxonomy and analysis of privacy preservation techniques for the IoT ecosystem along with the privacy threats addressed by each solution, their limitations, and their known resistance to attacks on user privacy. Chapter 4 showcases an approach for generating realistic synthetic and privacy-preserving smart health care datasets with fine-grained nutritional and activity user profiles by using GANs. In Chapter 5, we explore the domain of machine learning with privacy preservation on the resource-limited IoT devices, and propose and present a novel approach with corresponding algorithms and an end-to-end system pipeline for reconfigurable[1] data privacy in machine learning on resource-limited computing devices. In Chapter 6 we present the design and implementation of an end-to-end pipeline for time-series forecasting of health data streams in a federated learning environment. We also integrate state-of-the-art privacy preservation solutions in the designed pipeline and evaluate their impact on the time-series forecasting. Chapter 7 presents a differential privacy library *PyDPLib* for computing private statistics with medical data as a use case. Finally, we present our conclusions and future work in Chapter 8.

[1]We use the words reconfigurable and tunable data privacy as synonyms.

CHAPTER

2

Background

This chapter presents the necessary background by first introducing privacy preservation by design and default as specified in GDPR. Afterwards, this chapter overviews the generative adversarial networks used for synthetic data generation, followed by an overview of the privacy preservation solutions employed in this work such as differential privacy, anonymization and federated learning. Finally, we provide some background information on technologies used for time-series forecasting.

2.1 Privacy by Design and Default

In accordance with the EU's GDPR, all services employing any collection or usage of user data must provide data protection by design and by default [13]. Here The term "Privacy by Design" means nothing more than "data protection through technology design" [13]. This implies that by default the system should only collect and process data that is absolutely necessary as well as provide strict data protection guarantees. In the context of resource constrained devices as discussed in Chapter 5, this implies that it is vital to determine the most optimal set of features to be made private in order to provide the strictest possible privacy preservation on user data.

2.2 Generative Adversarial Networks

Generative adversarial networks (GANs) [28] is a unique kind of generative architecture that is inspired by the zero-sum game in game theory. It consists of two deep learning models, a generator and a discriminator, trained against each other as shown in Fig. 2.1. The goal of the generator is to capture or learn the distribution of the actual data and generate new data samples. The discriminator aims to detect whether the data is coming from the actual data distribution or is it a fake one generated by the generator, hence acting as a binary classifier. The two models compete with each other to improve their performance until they reach a Nash equilibrium where the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

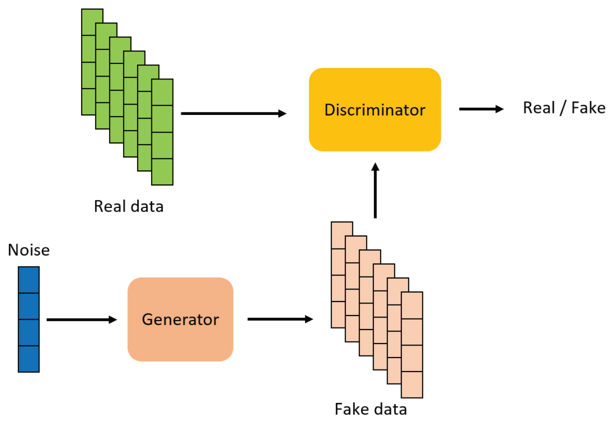


Figure 2.1: GAN model for synthetic data generation.

The GAN works as follows: Let G and D be differentiable functions that represent the generator and the discriminator respectively. G takes random variable $\mathbf{z}$ as input and generates a data record $G(\mathbf{z})$ and learns the distribution p_g over data $\mathbf{x}$ with a prior on input noise variables $p_{\mathbf{z}}(\mathbf{z})$. The generated record is then fed to D which also receives the real data record $\mathbf{x}$ from real data distribution $p_{data}(\mathbf{x})$ and tests for their authenticity. The discriminator, when shown both the $\mathbf{x}$ and $G(\mathbf{z})$ data, assigns probabilities $D(\mathbf{x})$ to the record where 1 represents a prediction of the record coming from the real data distribution and 0 represents the data as fake. With time, the discriminator D is trained to maximize the probability of assigning correct labels to both the training examples and the samples generated by G. G is trained simultaneously to minimize the $\log(1 - D(G(\mathbf{z})))$. In short, D and G play a two-player minimax with value function $V(G, D)$ given by [28]:

$$\min_G \max_D V(D, G) = E_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + E_{\mathbf{z} \sim p_{\mathbf{z}(\mathbf{z})}}[\log(1 - D(G(\mathbf{z})))]. \tag{2.1}$$

As shown in [28], the optimal discriminator $D^*_G(\mathbf{x})$ is given by:

$$D^*_G(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}. \tag{2.2}$$

Boundary-seeking GAN. Rearranging the equation (2.2) we get:

$$p_{data}(\mathbf{x}) = p_g(\mathbf{x})\frac{D_G^*(\mathbf{x})}{1 - D_G^*(\mathbf{x})}. \tag{2.3}$$

From the above equation we can see that even if G is not optimal, the true data distribution can still be found by scaling $p_g(\mathbf{x})$. Furthermore, the optimal generator $p_{data}(\mathbf{x}) = p_g(\mathbf{x})$ can also be obtained by making the discriminator ratio equal to 1, which means that $D(\mathbf{x})$ must be equal to 0.5 and then $D(\mathbf{x}) = 0.5$ is nothing but the decision boundary. For a perfect G, $D(\mathbf{x})$ cannot differentiate between real and fake data, or the real and the fake data are equally likely. Since $D(\mathbf{x})$ has two outputs, each with probability of 0.5, the objective function of G can be modified to force the discriminator outputting 0.5 for every generated data. This can be achieved by minimizing the distance between $D(\mathbf{x})$ and $1 - D(\mathbf{x})$ for all $\mathbf{x}$. Since $D(\mathbf{x})$ is the probability function, the minimum will be achieved at $D(\mathbf{x}) = 1 - D(\mathbf{x}) = 0.5$ and hence the generator G loss is given as [29]:

$$\min_G \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}\left[\frac{1}{2}(\log D(\mathbf{x}) - \log(1 - D(\mathbf{x})))^2\right]. \tag{2.4}$$

We use the Boundary-seeking GAN in our proposed approach for generating synthetic private smart health care data as it offers stable and efficient training.

2.3 Differential Privacy

Differential privacy (DP) is a privacy preservation technique that aims to maximize the accuracy of queries from statistical databases while minimizing the chances of identifying the underlying data records. Differential Privacy offers a provable and quantifiable amount of privacy protection by means of a privacy-loss budget ε. The most popular mathematical tool used to express DP is as follows [30]: A randomized algorithm A is ε-differentially private (ε-DP), if for the set of all datasets D and D' that differ on at most one row (i.e. the data of one individual), and any subset $S \subseteq range(A)$,

$$\Pr[A(D) \in S] \leq e^{\varepsilon}\Pr[A(D') \in S]. \tag{2.5}$$

The loss of privacy is quantified using ε, which is used to determine the noise addition for ensuring DP. As can be noted, absolute privacy is obtained when $\varepsilon = 0$, where the inclusion of an individual's data has almost no impact on the output of the randomized algorithm A. Achieving higher levels of privacy preservation (small ε) involves adding more noise to the data which leads to a decrease in the output accuracy of the algorithm and vise versa. In other words, decreasing the parameter ε means increasing the output accuracy at the cost of loss of privacy.

Therefore, a trade-off must be found between keeping the information private and achieving meaningful results, depending on the data and nature of the randomized algorithm. When an algorithm requires multiple additive noise mechanisms, the privacy guarantee follows from the basic composition theorem [31, 32] or from advanced composition theorems and their extensions [33–35].

Post-Processing Theorem in DP. The post-processing theorem in DP [4] states:

Theorem 2.3.1. If a mechanism M satisfies ε-DP, and g be any function, then $g(M(X))$ also satisfies ε-DP.

DP mechanisms leverage this theorem as they mostly focus on perturbing the distribution by noise addition. This perturbation is done either on the input data points or on the output of the querying statistical function. Applying the post-processing theorem, any data drawn from a noisy distribution that satisfies DP will also be ε-DP. We leverage this theorem to provide differentially private visual representations of statistics on protected health information.

Laplacian Differential Privacy. The Laplacian mechanism is one of the most popular noise addition mechanisms in DP [36]. The probability density function of Laplace distribution is shown in Figure 2.2 [1]. A standard approach is adding random noise with the Laplacian distribution proportional to the sensitivity S_f of the queried function to ensure DP-queries. Random noise is drawn from a Laplacian distribution with mean 0 and variance S_f/ε to achieve ε-differential privacy [4]. Mechanisms employing different privacy with Laplacian noise addition are referred to as the Laplacian Differential Privacy mechanisms in this thesis.

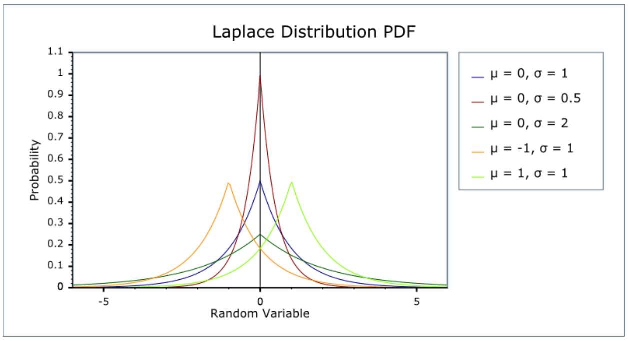


Figure 2.2: Laplace Distribution Probability Distribution Function.

[1]Source: `https://valelab4.ucsf.edu/svn/3rdpartypublic/boost/libs/math/doc/sf_and_dist/graphs/laplace_pdf.png`

Sensitivity. According to [4], sensitivity S_f captures the magnitude by which a single individual's data can change the output of the function f in the worst case. It helps to quantify the uncertainty in the response that needs to be introduced in order to hide the participation of a single individual. Mathematically, for any function f over the set of all datasets D and D' that differ on at most one row,

$$S_f = \max||f(D) - f(D')|| \tag{2.6}$$

where ||.|| denotes Manhattan distance or L1 norm [30]. We use Laplacian differential privacy by adding noise directly to the data records (with aggregated or non aggregated data points) in order to ensure easy integration into any data analytics system using any data visualization software. Each data point x is individually noised as x' by picking a random noise sample from a Laplacian distribution given by:

$$x' = x + Lap(0, \frac{S_f}{\varepsilon}) \tag{2.7}$$

where S_f represents the sensitivity of the statistical query.

2.4 Anonymization Techniques

Anonymization is one of the easy to integrate privacy preservation techniques and also offers the low computational complexity as compared to other techniques. Anonymization techniques commonly employ the principles of generalization and suppression. *Generalization* implies replacing a value with a less specific but semantically consistent value, while *suppression* involves not releasing a value at all [37]. Common practice in anonymization includes removal or modification of sensitive attributes such as names, gender, postal codes, and identification numbers. More sophisticated methods such as *k*-anonymization [7, 37], *l*-diversity [8] and *t*-closeness [9], are employed for better privacy preservation guarantees.

2.5 Federated Learning

Federated learning (FL) is a novel approach in distributed machine learning with two highly appealing characteristics: the gains in privacy and performance [38, 39].

The FL mechanism is shown in Figure 2.3. The FL mechanism learns from all participants' data without actually seeing it. The actors of the FL algorithm are the *clients* and a *central coordinator* (often called the *server*). Each client holds a local dataset which contains only their data. The server shares a central model with all the clients. Then, each client improves the current model using information from their local data and sends the update back to the server. The server aggregates the updates from multiple users and an improved central model is created and shared with the clients. The process repeats as the clients' devices collect more data.

FL caters to a variety of features suited to distributed ML on mobile client devices, such as catering to non-independent and identically distributed data, unbalanced and massively distributed datasets, and high capability to function in scenarios with limited communication.

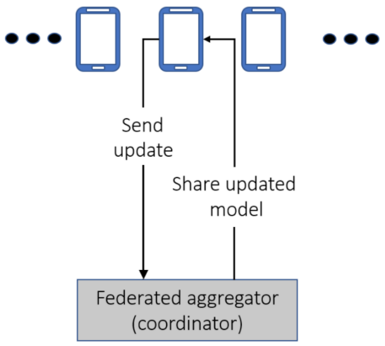


Figure 2.3: Federated Learning.

2.6 Time-Series Forecasting

We now provide a brief background on technologies used in time-series forecasting.

Time-series data. Time-series are sequences of observations ordered by some temporal information. Time-series analysis is done to extract meaningful trends in data. It has applications in numerous fields such as recommender systems, personalized shopping, or targeted advertising. Time-series forecasting is an active field of research since it plays a fundamental role in the decision-making process. For example, data collected from various traffic sensors can help predict traffic conditions to help the drivers avoid traffic jam in future [40]. Time-series analysis is helpful in various domains, such as signal detection, anomaly detection, classification, query, clustering, and forecasting.

In this thesis, we focus on time-series forecasting and discuss how time-series clustering can improve the predictions. More specifically, we exploit the similarities between time-series to build better models. We use a raw-data-based approach [41] for time-series clustering in our work as it is best suited to the requirements of the health care use case, as this approach discovers groups of users with similar time-series directly from raw data.

Distributed k-means algorithm. The k-means algorithm works by finding *k* centroids corresponding to *k* clusters such that the distance between each point and its closest center gets minimized. In this thesis, we will focus on the scalable variant of k-means for big datasets that parallelize the algorithm by distributing the computation to multiple workers and provide a good approximation of the solution. [42] tackles the problem by proposing a parallel k-means algorithm based on the Single Program Multiple Data (SPMD) model, using message-passing. Our implementation adapts the SPMD message passing model [42] to work on the streaming data.

Pattern matching. In time-series clustering, the pattern matching process is employed to discover groups of series that exhibit similar patterns. In the case of symbols sequences, the simplest method is to compare each symbol of the series at a given time in pairs. Techniques such as Hamming and Levenshtein distance are mostly used for measuring distance. We cluster the users' time-series to capture similar trends in user data using Hamming distance.

Neural Networks for Time-Series Data. Recurrent Neural Networks (RNNs) are particularly employed for sequential data as their output is constructed using both the input and the previous state. The previous state of the network is referred to as memory and is the key element in the architecture of an RNN. As RNN output is constructed using both the input and the previous state, the update rule is a recurrence relation:

$$h^t = f(h^{(t-1)}; x^t; \theta)$$

where h^t is the hidden state at step t which is a function f of the previous hidden state $h^{(t-1)}$ and the current input x^t, and θ represents the parameters of the function. It can be noted, the internal state at step t depends on the internal state at step $t-1$, which in turn depends on the internal state at step $t-2$ and so on. These loops allow the information to persist inside the network, meaning that the output will use historic data for each prediction. The RNNs are very effective when the prediction requires only information from the recent past. However, when the needed information for a prediction is at a considerable distance in the past, RNNs do not perform very well the gradient of the loss functions decays drastically over time, causing the long-term dependencies to be lost [43]. A variant of RNNs, Long-Short Term Memory network, is used to solve the vanishing gradient issue.

Long Short-Term Memory networks (LSTMs) [44] are well suited for learning order dependence in sequence prediction or classification problems including text [45] and speech recognition, anomaly detection, and time-series data forecasting [40].

The only difference between LSTMs and Recurrent Neural Networks (RNNs) is the computation of the hidden state of the network. While in RNNs hidden state is represented by only one vector, in LSTMs the hidden state is split into two vectors $h^{(t)}$ and $c^{(t)}$, which act as a short-term state and a long-term state, respectively.

The main idea of LSTMs is making the network decide which information is relevant and which information can be forgotten. Three special gates control which information is kept or forgotten at each step: the forget gate which decides which information should be thrown away, the input gate determines what new input information x^t is useful and should be added to the state, and the output gate builds the output. We use LSTMs for time-series forecasting in this thesis, as explained later in Chapter 6.

CHAPTER 3

The Case of Privacy in the IoT Ecosystem

3.1 Introduction

With the increasing popularity of the Internet of Things (IoT), the past decade has seen the appearance of a plethora of smart devices. According to Cisco's Visual Networking Index (VNI) 2021 forecast report, there will be 3.5 networked devices for every human on Earth by the end of 2021 [46]. Broadly speaking, any sensor that is capable of collecting data, processing it using built-in circuitry and transmitting it qualifies as a smart device. Typically, these devices upload the data to the cloud where it is further processed and stored in order to offer personalized services to the end users. An advanced variant of this approach includes offloading further processing and analytic capabilities to the devices, commonly referred to as *edge computing*.

Several models for the architecture of the IoT have been proposed in literature. Figure 3.1 shows a widely accepted general model with three layers [47]:

- L1: *perception layer* – consisting of the sensory devices collecting data.
- L2: *network layer* – responsible for collecting, aggregating, processing, and transmitting the data from the perception layer.
- L3: *application layer* – consisting of all the applications and solutions driven by the sensory data that are available to the users.

Based on this three-layer model, Chen [48] proposed that the IoT ecosystem is composed of four major components: *sensors* (in L1), *communication* (in L2), *computation* (in L2) and *service* (in L3). We will use both models interchangeably in this work.

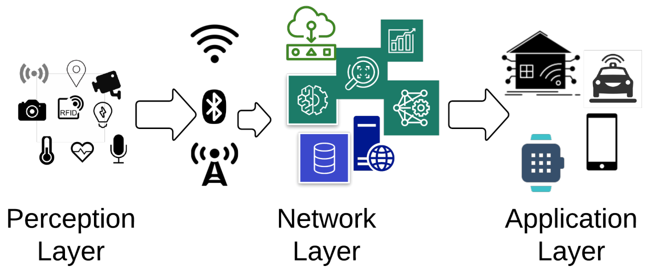


Figure 3.1: IoT Architecture Layers.

For example, a *wearable sensor*, such as a fitness tracker, is part of L1. The network infrastructure as well as the supporting technologies that store, aggregate and process this data (commonly in the cloud) are part of L2. Finally, users interact with fitness applications using their smart phones in L3.

Obviously, privacy is a major concern in IoT. In our above example, the fitness tracker collects information about the user's location with respective timestamp, heart rate, daily activities, etc. This data is then collectively analyzed by recommender systems to give users personalized health advises. In many cases, such recommender systems are driven by models created by machine learning (ML) algorithms. Unfortunately, these models are often sensitive towards specific training samples: Due to the nature of the datasets and the uniqueness of the data points, some of the training samples are implicitly memorized [49,50]. Research has shown that it is possible for attackers to replicate or recover the details of the recommender's underlying model, referred to as *model stealing* [51–53]. Moreover, private and sensitive training data can be recovered from the models by performing *model inversion* attacks [50,54,55].

When it comes to IoT devices and solutions available commercially, privacy is often confused with security, and secure solutions are often marketed as privacy preserving. Moreover, existing solutions and techniques mainly focus on securing the communication channel as well as authentication and authorization mechanisms. Much less consideration is given to the preservation of privacy in the data collection, aggregation, storage and retrieval processes [56]. There is an imminent need to introduce privacy in all components, which requires better understanding of privacy threats in the IoT ecosystem. Furthermore, it is important to analyze the impact of privacy preservation techniques on data analysis and quality of service in terms of trade-offs between accuracy, privacy and efficiency.

This chapter presents an overview of privacy preserving techniques for IoT along with the privacy threats addressed by each solution, their limitations, and known resistance to attacks on user privacy. For this work, we focus on IoT devices and services used for personal applications such as health care and smart home solutions. Moreover, we focus on the three components *sensors*, *computation* and *service* since, since they have received much less attention so far than the *communication* component, as mentioned above. Consequently, we consider privacy in communication protocols to be outside the scope of this work.

Contributions: Our main contributions can be summarized as follows:

- We propose and present a taxonomy of privacy preserving techniques and solutions for IoT ecosystem;
- We provide a comparison of privacy preserving techniques and solutions that we have observed in this work;
- We analyze the techniques in the light of the EU's General Data Protection Regulation (GDPR);
- We identify some open issues in privacy preserving techniques that should be addressed.

3.2 Privacy Threats and Attacks in IoT Ecosystem

According to [57], privacy is "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others". A definition of privacy concerns is proposed by Smith et al. [58]: concerns for collection of personal information, concerns for unauthorized secondary use (internally in organizations and externally), concerns for improper (unauthorized) access to personal data and concerns for errors in collected personal information.

In order to categorize privacy preservation solutions, we first identify privacy threats in the IoT ecosystem and the architecture layers associated with them. Afterwards, we provide an overview on attacks on privacy and the respective threats associated with them.

3.2.1 Privacy threats

Ziegeldorf et al. [59] categorize the most common privacy threats in IoT. In the following, we give a short overview on the threats and we attribute them to the different layers of the IoT architecture. Note that the threats often occur in combination in IoT solutions, depending on the type of service offered.

3.2.1.1 Identification

Denotes the threat of associating a persistent identifier with an individual or their data. For example, a name, pseudonym, an image or voice, or an address can be associated with an individual from a database or collection. Classified as the most common threat.
Affects: information processing in the network layer (L2).

3.2.1.2 Localization and tracking

With a notion of identification, this denotes the threat of determining an individual's physical location and recording it over time without authorization or consent. Location based services (LBS) commonly suffer from this threat as they can enable GPS stalking [60], though internet traffic can also be exploited for this purpose. Moreover, IoT-based LBS in indoor environments require additional constraints on data sharing and authorization, e.g., they can enable stalking and unintended bias in work environments.
Affects: all layers of IoT architecture, though it is more visible on the network (L2) and application layer (L3).

3.2.1.3 Profiling

Users are profiled for the sake of personalization but this often results in unwanted advertisements, price discrimination or biased automatic decisions. In an IoT-based environment, this threat is more imminent due to the availability of multiple information sources which potentially allow compiling complete information dossiers about individuals and inferring user preferences by correlation with other profiles.
Affects: information processing in the network layer (L2), especially in scenarios that require data dissemination or sharing with third parties.

3.2.1.4 Interaction and presentation

Similar to shoulder surfing, this refers to the threat of violating user privacy by conveying some private information intended for a specific user over a public medium. For example, one may get recommendations through the speaker or screen of a smart thing and people in the vicinity can also observe that information.
Affects: application layer (L3). Also occurs on the perception layer if the solutions offered are presented using peripherals of the sensory devices (e.g., speaker or screen).

3.2.1.5 Lifecycle transitions

This threat occurs upon change of ownership of the IoT devices. Most IoT devices are sold with the assumption of buy-once-use-forever, and log huge amounts of personal data throughout their lifetime history. This data (and its impact on personalization offered by the IoT device) may not be completely removed upon a memory wipe before transfer of device ownership.
Affects: information collection in the network layer (L2).

3.2.1.6 Inventory attacks

This threat mainly occurs due to communication capability of the sensor devices, which enables unauthorized access or collection of data. Unauthorized parties can also observe the communication pattern (and other distinguishable properties) and deduce the presence of devices as well as their type and model. Moreover, inventories can give information on user preferences which may be exploited by law enforcement agencies to conduct unwarranted searches or by burglars for targeted break-ins.
Affects: information collection in the network layer (L2). May be enabled using application layer (L3) by exploiting security flaws in the application (L3) or perception layer (L1).

3.2.1.7 Linkage

Users consent to sharing different attributes of personal data with each IoT service they use. However, the combination of data collected from independent sources can reveal information about individuals that they originally did not consent to reveal [61]. Moreover, data maybe be incorrectly inferred due to loss of context as a result of the combination of different permissions (data access restrictions).
Affects: Information dissemination in the network layer (L2).

3.2.2 Attacks on user privacy

Here, we briefly describe some of the common attacks on user privacy. Note that it is not an exhaustive list of possible privacy attacks. Descriptions of more attacks targeting IoT ecosystem components (e.g. databases and ML models) that in turn compromise user privacy, can be found in [62, 63] and other works.

3.2.2.1 Membership inference attack

With this attack, the adversary can reveal whether or not a specific data record was used to train the ML model, given that the adversary has knowledge of the ML model and the individual data record [64, 65]. Privacy is violated in this attack if inclusion of an individual in a training set is itself sensitive. For example, inclusion as a data record in a health-related ML model leaks information about the health of that individual. In terms of privacy threats, this attack directly threatens the identification of a person, can aid in profiling, and can make use of linkage and inventory attacks.

3.2.2.2 Data inference attack

As observed by [66], this attack is commonly associated with encryption-based privacy preserving solutions. It tries to recover some information about a given data record by using Linkage (with publicly known information) and making tailored queries to the system and observing the responses to see if any information about underlying records is leaked. A classical example of this attack is using frequency analysis to break ciphers.

3.2.2.3 Attribute disclosure

Attribute disclosure occurs when some released data records make it possible to infer characteristics of an individual more accurately than is generally known about that individual [9]. In other words, new information about some individuals is revealed by the data release. This attack commonly uses linkage from multiple data sources to infer user information.

3.2.2.4 Fingerprinting and Impersonation attacks

Using Inventory attacks, an adversary might observe the communication pattern of a device and try to mimic it [67, 68]. If the privacy is compromised, the adversary might be able to access credentials of the device to alter privacy preferences of the user and inject fake data into the system.

3.2.2.5 Re-identification attacks

In this attack, an adversary can use linkage to combine data from multiple collections to re-identify a record from outsourced, published or open data records [69]. Re-identification is a very commonly observed attack, with the classic example of a voter list used for re-identification of a government official's health record from the records released by a health insurance company in 1997 [70].

3.2.2.6 Database reconstruction attacks

As observed in [71], confidential data may be vulnerable to database reconstruction attacks when statistical data is published by agencies for research or information purposes. This enables partial or full reconstruction of the original database records, which may lead to identification or unintended profiling of some users based on their association with some attributes in the targeted database.

3.2.2.7 Model stealing

Similar to database reconstruction, it is also possible to reconstruct or reveal the internal training parameters and other sensitive details of ML or recommender models using model stealing techniques [51–53,72]. This reveals sensitive information about the training data used for these models and can result in unintended profiling of users.

3.2.2.8 Model inversion

By observing ML model predictions, model inversion attacks enable adversaries to extract underlying training data of the individuals, as observed in [50,54,55]. A specific training record may not always be extracted as a result of this attack. Instead, the adversary will extract an average representation of inputs that are similarly classified. However, this can be a huge privacy threat if the exposed classes are sparsely populated, i.e., a class may correspond to a single individual in the records [54].

3.3 Taxonomy of Privacy Preserving Techniques

We present a taxonomy of privacy preserving techniques that eliminate the risk of privacy threats (presented in Section 3.2.1) and prevent the attacks on user privacy (described in Section 3.2.2).

Terminologies

Techniques represent the general principle(s) and methodology employed for privacy preservation, e.g., anonymization. *Solutions* represent algorithms designed using these principles. *Functional limitations* refer to design limitations on where the solutions can be applied depending on the data or the nature of the algorithm. *Non-functional limitations* include issues such as performance, scalability and accuracy.

3.3.1 Anonymization techniques

The industry and health care sectors have been employing data de-identification for years as a privacy preserving measure [73–75]. Common practice includes removal of some sensitive attributes like names, gender, state codes, or identification numbers – commonly referred to as *personally identifiable information (PII)*. Moreover, more sophisticated methods such as k-anonymization [7,37] and l-diversity [8] and t-closeness [9], are employed for better privacy preservation guarantees.

3.3.1.1 *k*-anonymity

k-anonymity provides privacy protection by guaranteeing anonymity between *k* entries – each released data record will relate to at least *k* individuals in the collection even if the records are directly linked to external information [7,37]. It uses generalization (replacing or re-coding a value with less specific but semantically consistent value) and suppression of records (not releasing a value at all) to achieve privacy goals. However, these might skew the characteristics of the original dataset. Functional limitations include data diversification to ensure there are at least *k* similar records in the database. *k*-anonymity has been shown to perform well for location based services (LBS) to prevent fake data injection attacks [76] and for privacy-preserving publishing of Electronic Health Records (EHR) [77]. However, it has been shown that *k*-anonymity is susceptible to data inference attacks [78], as well as attribute disclosure [9], re-identification attacks and database reconstruction attacks [79]. Improved versions such as (α, k)-anonymity model [79], have been proposed in literature to mitigate re-identification and database reconstruction attacks.

3.3.1.2 *l*-diversity

This solution improves upon *k*-anonymity and provides protection against attribute inference attacks [8]. Each anonymized group of (generally *k*) users has at least *l* "well represented" sensitive attribute (SA) values. Another improved version requires to have at least *l* distinct SA values in each group, called *distinct l-diversity* [9]. Similar to *k*-anonymity, functional limitations include diversification in dataset, as we need to ensure presence of well distributed SA values. However, in some cases, attackers are still able to associate an individual's record to have a certain SA when that value appears more frequently than other values in the group [77].

3.3.1.3 *t*-closeness

This solution improves upon its precedents and aims at limiting the distance between the probability distributions of SA values within an anonymized group and SA values in the entire dataset [9]. This method provides better privacy guarantees against attribute disclosure as the attacker can not learn any information about an individual's SA value other than what is already available from the entire dataset. Some practical implementations have found *t*-closeness to be resistant to attribute disclosure attacks, however, its resistance to membership inference attacks still needs to be investigated [80].

Researchers have also combined these solutions for better privacy guarantees. For example, Yin et al. [81] propose using *k*-anonymity and *l*-diversity in combination to prevent imbalanced sensitive attribute distribution in datasets to prevent attribute disclosure attacks. Moreover, there are many versions of all the aforementioned techniques proposed in literature, each focusing on protecting against a specific type of attack depending on the use case.

3.3.2 Model or output obfuscation techniques

User re-identification by model inversion attacks can be prevented by obfuscating the output of ML models within a provided range. Differential privacy is a solution that aims to maximize the accuracy of queries from statistical databases while minimizing the chances of identifying its records [4]. A function ensuring ε-differential privacy adds appropriately chosen random noise (with Laplacian distribution) to the true answer of an ML model to produce the response. This implies a fixed uncertainty in all measurements, implicating less probability of exposing a specific record. However, differential privacy alone cannot provide privacy guarantees for all scenarios due to certain functional limitations: a) it is designed for low sensitivity data queries, and b) using statistical inference and adaptive querying, one can infer the form of the underlying data distribution.

Differential privacy can be regarded as the most widely researched and adopted solution for privacy preservation in the current era. It is highly effective against model inversion and inference attacks, and is being used heavily in combination with other techniques to develop privacy preserving applications and services [49, 64, 82–84].

3.3.3 Multi-tier Machine Learning as a privacy preservation mechanism

Training openly available ML models on sensitive user data directly allows for data memorization. This technique proposes introduction of multiple training levels, which can reduce the footprint of distinct and sensitive training data on output models. Semi-supervised knowledge aggregation and transfer [49] is a multi-tier ML solution that proposes a teacher and student models hierarchy. Teacher models train directly on partitions of sensitive data, and afterwards apply an aggregation mechanism as privacy preserving layer to train a student (openly available) model on non-sensitive data using the teacher models. This technique uses differential privacy to define privacy-preserving properties of student models during the training phase. As only student models are published, using model inversion attacks cannot compromise the original training samples given the fact that noisy voting is used in the training procedure instead of considering the absolute majority of classification decisions of teacher models. This is a relatively new distributed solution with strong privacy guarantees and applicable to a wide range of ML models. However, its utility with respect to quality of recommendations needs to be researched.

3.3.4 Decentralized machine learning

Decentralized machine learning solutions offer a new computing paradigm for better privacy preservation. Instead of transmitting (potentially sensitive) user data to computation, a part of the computation is offloaded to end-user devices and each device contributes partial updates to the system model. Doing so eliminates the risk of exposing sensitive and private raw data to the service provider as well as other *honest-but-curious* adversaries present in the environment. Federated machine learning [85,86] has become an increasingly popular solution based on this technique in the past few years and is increasingly being researched and used in ML models and recommender systems [87–90]. It proposes creation of a global model as a result of learning attributes from updates pushed by users. Since it is a relatively new technique, it caters well to the nature of distributed computing systems used in the IoT ecosystem – it is highly scalable and efficient – although there is a need to investigate how different applications and use cases can be ported to this solution. Federated machine learning can, however, be susceptible to inference attacks [91] as it exposes intermediate results which may actually leak important information about user data [88].

3.3.5 Cryptography-based solutions

It is believed that if data is encrypted during analysis, user privacy can not be compromised. Homomorphic Encryption [92] is a cryptographic solution that allows computation to be executed directly on encrypted data. It supports addition, multiplication, and quadratic functions. Moreover, homomorphic encryption offers privacy-preserving capabilities in both training and classification phases of ML models, unlike most of the existing works that only focus on the training phase. Homomorphic schemes are further classified as fully or partially homomorphic.

3.3.5.1 Partially or Somewhat Homomorphic schemes

These solutions support limited operations like addition and multiplication as well as other operations on ciphertexts, but do not support arbitrary computation on ciphertexts. These schemes perform relatively well in practice and have better performance due to lower computational complexity as compared to fully homomorphic schemes [93]. However, fewer algorithms can be implemented using the restricted set of operations [6].

3.3.5.2 Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) schemes not only support multiplication and addition, but also support quadratic function and arbitrary computation on ciphertexts. Classifiers designed using this schema are privacy preserving by nature and are better suited for real world applications in terms of privacy guarantees, because they support arbitrary computation. However, few fully homomorphic encryption schemes exist, and they are often computationally expensive, i.e., around 2-5 seconds per operation [6]. Some efficient FHE schemes have been proposed [94], but they have been found susceptible to data inference attacks like encryption key recovery and data decryption in both known message (broadcast) and unknown message (secret) scenarios [95].

Other popular solutions include garbled circuits and Secure Multi-Party (SMP) computation protocols. Originally proposed by Yao in the 1980s [96] as a secure way of computation, garbled circuits are now used extensively for providing privacy guarantees. Similarly, SMP solutions are also being investigated for providing privacy guarantees in information processing.

3.3.6 Data summarization techniques

Exposing raw user data not only poses communication overhead but also puts user privacy at risk. This technique proposes creation of aggregated and summarized versions of datasets for efficient creation of ML models as well as providing user privacy. This poses the trade-off between accuracy of data and privacy preservation. The data summarization solution proposed by [97] uses this technique for improving performance and potentially enhancing privacy preservation in recommender systems. It marks portions of the data as private and summarizes the rest of the data from all users to create a representative training dataset. Further implementations have combined the use of data summarization with differential privacy for stronger privacy preservation [98].

When it comes to non-functional limitations, similar to decentralized ML, data summarization is a relatively new technique, is scalable and efficient to cater to the nature of distributed platforms and systems used by IoT services. However, the privacy guarantees achieved by using this solution need to be investigated.

3.3.7 Ensuring privacy with dataflow models

This technique proposes creation of data flow models with respective permissions at each level to ensure user privacy and transparent accountability.

3.3.7.1 Blockchain to ensure privacy and verifiability

Researchers have proposed the use of blockchain for verifiability and accountability of data collection, storage and access in IoT environments [99–102]. For example, blockchain-based data provenance can provide tamper-proof records and enable data accountability in the cloud [100]. Moreover, blockchains are being extended for use in the context of IoT for healthcare, as surveyed in [103]. However, there is room for research for introducing scalability in blockchains so they can adapt well to IoT environments.

3.3.7.2 Privacy-centric programming languages and platforms

These solutions require information flows and privileges to be declared beforehand, so all the data elements are attached to respective policies [67, 104–106]. For example, Jeeves [106] is a privacy centric programming language, used as an add-on library with Java. HomePad [104] applications are implemented as directed graphs of elements (instances of functions that process data in isolation). It allows for automatic verification of the application's flow graph against user-defined privacy policies with low computational overhead by modeling these elements and the information flow graph. In addition to that, [107] outlines some guidelines for privacy preservation while designing IoT applications.

3.3.8 Personalized data stores

Personalized data stores offer a flipped environment for privacy preservation, where the users collect and maintain their data from multiple sources in one place, e.g. an encrypted data store, and authorize its informed use. The Hub-of-All-Things project (HAT) is a solution that proposes total control of data by the user and monetizing this data [108, 109]. Instead of storing data on different platforms, it is aggregated in the data store and users can offer their data to interested parties in exchange for personalized services.

3.3.9 Privacy preservation at processing level

This technique proposes secure and private compute/processing units to ensure that no data or computation is exposed in the entire information flow. Intel® introduced Software Guard Extensions (SGX) [10] as a solution that proposes the use of "enclaves", protected areas of execution, to protect selected code and data from disclosure or modification. A huge merit of this solution is that it is a hardware-assisted execution environment with the smallest possible attack surface: the CPU boundary. It also provides specific architecture instructions to mark portions of data and code as private, which makes it similar to sandbox concepts in the security domain. In principle, it is a privacy-preservation solution for both users and corporations – users may execute analytic codes locally without moving their data anywhere, and corporations can analyze data on user-end without exposure of their algorithms. However, recent research has shown that it is susceptible to some data inference attacks using side-channel information like cache-timing [110] when working with weaker versions of encryption algorithms. It also requires designing application complying with a specific programming model, which may be inefficient for adapting private implementations of algorithms currently in use by large organizations.

Table 3.1 summarizes the results of our analysis and classification of privacy preservation techniques and solutions, affected IoT layers and their known resistance towards attacks. For each of the privacy preservation solutions in the table, we indicate a level of privacy, namely strong/mediocre/weak resistance, based on the assessments provided in the literature (for selected reviewed papers).

Table 3.1: Analysis of privacy preserving solutions

Privacy preserving technique	**Solution**	**Merits**	**Affected IoT layer**	**Relevant privacy threat(s)**	**Limitations**	**Trade -offs**	**Relevant attacks**	**Known resistance**
Anonymization	*k*-anonymity	Easy to implement, low complexity	L2 (information aggregation)	Identification, localization and tracking, profiling, linkage	Requires diverse data	accuracy/ privacy	Re-identification, Database reconstruction , Data inference, Attribute disclosure	Weak resistance (some implementations) [9,78]
	l-diversity	Low complexity	L2 (information processing)	same as above	Requires diverse data	accuracy/ privacy	Attribute disclosure	Mediocre resistance [9,77]
	t-closeness	Protects sensitive attributes	L2 (information processing)	same as above	Requires strong dataset diversification	accuracy/ privacy	Attribute disclosure	Strong resistance [80]
Model or output obfuscation	Differential privacy	Easy to integrate with solutions	L2, L3	Identification, profiling, linkage	Works for low sensitivity data queries	accuracy/ privacy	Model Inversion, Inference attacks	Strong resistance [64,82,84]
Multi-tier ML	Semi-supervised knowledge transfer	Distributed, applicable to any ML model	L2, L3	Profiling, linkage	Affect on accuracy of ML models is unknown	accuracy/ privacy	Model stealing and inversion, Inference attacks	Strong resistance [49]
Decentralized ML	Federated ML	Highly scalable and efficient	L2, L3	Inventory attacks, linkage, profiling	Potential information leakage	efficiency/ privacy	Fingerprinting and impersonation attacks, Inference attacks	Mediocre resistance [91]
Cryptography	Fully Homomorphic encryption	Private ML models training and classification	L2	inventory attacks	large computational overhead	efficiency/ privacy	Data inference (data/key recovery)	Strong/ mediocre resistance [95]
	Partially/ Somewhat Homomorphic encryption	Relatively lower computational overhead	L2	Inventory attacks	May not be applicable to all ML models	accuracy/ privacy	Inference attacks	Mediocre resistance [93]
Data summarization	Public-private data summarization	Highly efficient solution with minimal loss of accuracy	L2	Identification	Unquantified Privacy guarantees	accuracy/ privacy	Inference attacks	Unknown
Data flow models	blockchain for privacy	Verifiable privacy	L2	Inventory attacks	Computational overhead, low scalability	efficiency/ privacy	Fingerprinting and impersonation attacks	Strong resistance [100]
	privacy-preserving programming languages and platforms	Low overhead with verifiable privacy	L2, L3	Inventory attacks	Information flows to be declared beforehand	efficiency/ privacy (in some cases)	Fingerprinting and impersonation attacks (some cases)	Strong resistance [67]
Personalized data stores	HAT	User controls and monetizes her data	L1, L2	Linkage (under consent)	Requires users to pay for storage	cost/ privacy	Attribute disclosure	Unknown
Private Compute Units	Intel® SGX processors	Protected execution environment; no data or computation is exposed	L2 (information processing, computation)	Inventory attacks	Requires specific application design for SGX programming model	efficiency/ privacy	Data inference (using side-channel information and cache-timing [110])	Strong/ mediocre resistance [110]

3.4 Privacy-aware ML and Data Mining

A number of privacy preserving implementations of machine learning and data mining algorithms can be found in literature. Papernot et al. [111] survey the state of the art of privacy preserving ML algorithms. Moreover, differential privacy is used extensively in ML models for protection against model inversion attacks [49, 112–114].

Chiron [115] is an interesting implementation of privacy-preserving ML-as-a-service, designed particularly for cloud environments which form a major part of the IoT ecosystem. It uses private compute units (with SGX) to enhance privacy guarantees. Moreover, implementations of *k-anonymity* in combination with ML algorithms [116] and cryptography techniques with ML [117] also exist in literature.

When it comes to data mining, as mentioned in Section 3.3.4, various implementations of recommender systems use federated learning as a privacy preservation measure [87–90]. Collaborative filtering is used extensively in recommender systems [118]. Some privacy-preserving implementations include [119], which combines *k*-anonymity with collaborative filtering; [5], which applies obfuscation; and [120], which uses differential privacy in combination with homomorphic encryption to ensure private recommendations. Also, [89] proposes a federated ML version of collaborative filtering for personalized recommendations.

3.5 GDPR and its Implications

The GDPR law enforces all organizations that collect and process data from users to include Privacy by Design and Privacy by Default (originally explained in [121]). Privacy by Design dictates that organizations should design all their services involving processing of personal data while considering data protection and privacy measures at every step. Privacy by Default dictates that all public services should apply the strictest privacy settings by default, without any manual input from the end-user. The GDPR also grants some basic rights to end-users: right to (give and withdraw) consent, right to be forgotten and right to access (personal) information [122]. Veale et al. [123] analyze the impact of incorporating the GPDR law in ML models for protection against model inversion and membership inference attacks. They conclude that some ML models may need to be legally classified as personal data as a result of this law.

Relatively new privacy preserving techniques proposed in literature are GDPR-compliant by design. For example, personalized data stores are directly based on the principles of user consent and the right to access. The right to be forgotten can also be exercised by removing the data access from organizations that fail to comply with the user's privacy preferences. Also, for private compute units, since user data can potentially always stay on the device, the right to access data is respected.

However, organizations need informed consent of the users for analyzing their data. Similarly, data flow models (solutions using blockchain and pre-defined information flows) are also GDPR-compliant by design. Moreover, for these solutions, the user defines privacy preferences and is able to verify if they are respected by the service. For solutions based on data summarization, users may not be able to exercise their right to access information, as the information is used in a modified (summarized) form. Moreover, cryptography-based techniques may also hinder the right to access collected information although they may ensure privacy by design and by default. Other techniques based on multi-tier and decentralized ML might also not be able to comply with the right to access information as it might give out sensitive details about how the organizations are training their ML and recommender models. We believe that, in principle, it is hard to enforce the right to forget in ML algorithms once user data has already been used to train an ML model (though the effects of data point on the trained model might disappear eventually), which in turn implies that they should be classified as personal data as proposed by [123].

3.6 Open Issues and Future Work

In the light of our analysis of privacy preserving techniques and the discussion on GDPR presented above, we identify some open issues and suggestions for future work. First, it is advised to use synthetic or representative datasets for where exact computations are not needed [124]. Moreover, there is a need to find an optimal trade-off between data utility and privacy preservation when generating the representative datasets. Solutions for data summarization should be combined with other privacy preserving techniques for better privacy guarantees. However, the effect of combining different techniques on accuracy and efficiency of solutions needs to be investigated. Also, there is a strong need to formulate guidelines for publishing privacy preserving open datasets, ML and recommender models. Additionally, blockchains solutions might be a good candidate for verifiable privacy preservation on user-end. In general, there is no clear winner among the privacy preservation techniques – depending on the use case, some techniques will outperform others in terms of robustness towards attacks. Another interesting observation is that industry and healthcare organizations have often found the relatively weaker solutions to be strong candidates for privacy preservation.

3.7 Summary

In this chapter, we identified privacy threats on different layers of the IoT ecosystem as well as associated attacks on user privacy. We presented a taxonomy of state of the art privacy preservation techniques along with their limitations, susceptibility to privacy threats and their proved robustness towards attacks on privacy. There is no clear winner among the privacy preservation techniques. Instead, depending on the use case, some techniques will outperform others in terms of robustness towards attacks, or in terms of efficiency or better accuracy. Similarly, depending on the use case, model obfuscation techniques, multi-tier and decentralized ML, private compute units and data flow models using blockchains and pre-defined information flows emerge as relatively strong candidates for privacy preservation. However, not all of these solutions can guarantee the rights granted by the GDPR to users. Moreover, it is recommended to use synthetic representative datasets where exact computations are not needed. Solutions from different privacy preservation techniques should be combined for better guarantees on privacy preservation. Moreover, the resulting impact of combining privacy preservation techniques on the system accuracy and efficiency needs to be investigated.

CHAPTER

4

Synthetic and Private Smart Health Care Data Generation using GANs

4.1 Introduction

The Internet of Things (IoT) paradigm as we know it today is a fruition of the technological advancements in the area of computer networks and communication, that ensure the functionality of these services driven by highly interconnected components. The mass adoption of IoT devices and services creates a plethora of valuable data pools that have applications in areas such as smart health care [125], smart cities [126], smart farming [127] and personalized medicine [128]. These applications are often driven by machine learning (ML) algorithms which ensure the provision of continuously improving personalized services. However, ML-based algorithms and services require access to huge amounts of sensitive and private data, which might not always be reasonable and in some cases, impossible to obtain and share due to local data protection laws. In particular, the advancements in the health care sector are hindered due to the curse of limited data access.

Data access is limited mainly because of the presence of highly sensitive medical information that not only arises concerns for personal privacy but also the threat of misuse or re-identification. Data protection laws like the EU's General Data Protection Regulation (GDPR) ensure higher public trust in data sharing, and informed use of collected user data by the companies. Realistic synthetic datasets offer the benefits of a) enhanced user privacy with reduced risk of re-identification, b) reduced risk of exposure due to privacy-breaching attacks on ML models such as *model inversion* [54, 123], and c) removal of data that could potentially expose competitive advantage for the data providers; all while maintaining fidelity to the real-world data. Therefore, realistic synthetically generated datasets are poised to accelerate the technological advancements in ML, as these datasets do not suffer from the curse of limited availability and can facilitate wide-scale data sharing and usage by industry and researchers without privacy concerns [11,124,129–132].

Generative modeling is a popular way to model synthetic datasets. These models learn the probability distributions of the given data and are capable of generating very realistic sample distributions from the same data. Hence, generative models are commonly employed for synthetic data generation as well as data augmentation. Generative adversarial networks (GANs) [28] and their variants have recently become a widely adopted approach for synthetic dataset generation [29,131,133–136]. However, generating tabular data with GANs, particularly smart health care data, poses unique challenges [133]. The first challenge is the presence of mixed data types, as the real-world data contains both discrete and continuous variables. The second challenge is to accommodate static and behavioral data types. For example age, height and weight are considered static variables as compared to the activity data, as the latter has a higher frequency of recorded changes in observation. Moreover, the data distributions might not always be Gaussian, which makes them harder to normalize or model with GANs. Finally, the major challenge in real-world data comes from highly imbalanced categorical data, as the individuals may possess widely diverse categorical attributes. Moreover, the frequency of logged measurements differs from individual to individual.

GANs can also be combined with different privacy preservation solutions to ensure strong user privacy in the synthesized datasets. Differential privacy (DP) is one of the most popular solutions used in combination with GAN. This approach relies on noise addition to either the learning mechanism or directly to the data. Research shows several variants of differentially private GANs that employ the noisy learning mechanism [131,133,136–141]. These DP-strategies are also applied in combination with different variants of GANs depending on the use cases. Moreover, GANs are being extensively used to generate Electronic Health Records (EHRs) [129,131,140,142]. Esteban et al. [140] use a Recurrent Conditional GAN (RCGAN) to generate synthetic time-series EHRs with the noisy learning process. Similarly, Baowaly et al. [131] generate EHRs by using Wasserstein GAN with gradient penalty (WGAN) and boundary-seeking GANs (BGANs). Their evaluations show BGANs to be more suitable for EHR generation.

We address the problem of generating smart health care records using BGANs. Our smart health care dataset is not only more diverse in nature but also possesses the 3 V's of big data (*volume*, *velocity*, and *variety*) as compared to the EHRs. We also used WGANs in our initial set of experiments in comparison with BGANs. Our findings suggest that BGANs are more suitable for synthetic smart health care dataset generation, due to the faster convergence of the BGANs and higher quality of the generated dataset. Moreover, our proposed approach provides additional privacy preservation by integrating DP in different settings. Our results show that the proposed approach is able to generate realistic smart health care data samples with user privacy guarantees.

Our contributions can be summarized as follows:

- We collect and refine a real-world smart health care dataset from geographically distributed users.
- We augment the collected smart health care dataset to represent diverse nutritional and activity patterns based on age, ethnicity, geolocation, dietary preferences, and other factors.
- We propose a method based on GANs for generating synthetic and tabular time-series data, containing categorical and numerical values, as well as the methods to generate the privacy-preserving versions of the synthesized data samples.
- We have created realistic synthetic and privacy-preserving smart health care datasets with fine-grained nutritional and activity user profiles for open use in research.

4.2 Data Processing Pipeline

This section presents our method for data collection, imputation, and transformation. Moreover, we present our approaches for privacy-preserving model training, followed by the data inverse-transformation mechanism. The proposed pipeline is shown in Figure 4.1.

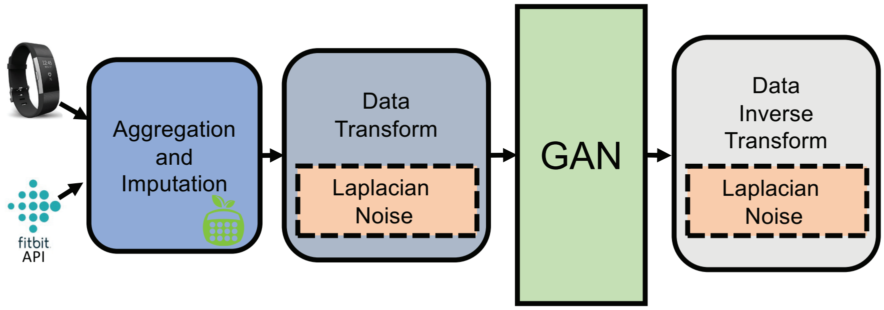


Figure 4.1: Data processing pipeline.

4.2.1 Data collection and imputation

For this work, we used Fitbit Charge 2 HR smartwatches for automated data collection in combination with the Fitbit App for manually logging user meals. A total number of 25 subjects were observed during this study, distributed across Belgium and Sweden. 12 devices were used for dataset collection, with 2 continual participants (male and female), and 10 users in circulation. The users were asked to record a minimum of 60 days of observations. The participants' pool consisted of 6 coarsely defined ethnicities to represent the overall health and diet patterns of the residing communities.

We collected more than 17M measurements related to the users' meal logs, calorie intake, heart rate, calories burned, steps taken, activity profile during the day, and sleep. Apart from the numeric data collection, the platform also collects categorical user data, such as age, gender, height, and weight. Since the users were not provided with smart scales, the weight measurement is logged manually. All these logs and measurements were then exported from the Fitbit platform. The collected data had many inconsistencies and user errors such as: 1) users forgetting to wear the watch on some days, 2) incorrectly recording very large portion sizes of meals, 3) manually recording the meals without caloric breakdown, and 4) incorrectly wearing the watch which resulted in inconsistencies between recorded activities, steps and (sometimes) heart rate. Moreover, some users logged their data using languages other than English, e.g., French or Italian. We spent huge amount of additional effort in translating the recorded meals with respective proportions to English, cleaning the user data, and identifying and fixing the inconsistencies in logged measurements. Afterwards, we aggregated and imputed the time-series data as follows.

4.2.1.1 Time-series data aggregation and imputation

Since this time-series data collection requires partial manual data logging, there are natural gaps in the time-series caused by these factors: users forgetting to wear the device or wearing incorrectly, users forgetting to log the meals, and meals not present in food database or customized meals with unavailable nutritional breakdown.

For the gaps in time-series, the days without any meal log entries were omitted. The remaining entries were analyzed for correctness in the recorded measurements. In case of a missing activity profile or a mismatch between the burned calories versus activity profile during that particular day, the user behavior pattern was analyzed to find the closest matching activity profile or the burned calories recorded in the past. A similar approach was used to remove the mismatch between the recorded resting heart rate (RHR) and the steps taken versus the activity profile.

4.2.1.2 Meal logs imputation

Imputation for missing nutritional breakdown for meals is more complex as compared to other missing attributes. When it comes to the available food databases from Fitbit, the United States (US) database is the most largely populated but specialized to the foods available in the US region. On the other hand, the Belgian (French) food database is partially populated. However, since there is no specialized database available for the users in Sweden, the users either recorded the closest matching entry in the US food database or in some cases, the users manually logged the nutritional breakdown for customized meals. A translation API was used to convert logs from other languages to English, replacing the log with either the closest match in the US food database or by using an external nutrition API. Nutritionix API [143] was used to impute the missing nutritional breakdown for meals. Initially, the measurements were aggregated into 3 records. Each contained the nutritional breakdown for a meal (breakfast/lunch/dinner), calories burned during the mealtime, RHR from the previous day, and steps taken as well as the activity records for that day. These records were later aggregated to form one record per day for the nutritional breakdown of all meals, activity profile, steps taken, calories burned, and RHR. The users exhibited all kinds of natural behavior, ranging from very sedentary to highly active users. The complete spectrum of data features (and ranges) is shown in Table 4.1.

Table 4.1: Dataset features with ranges (aggregated per day).

Features	**Type**	**Unit**	**Range**
Age	static	yrs	median: 28
Gender	static	-	0: male, 1: female
Height	static	cms	*private*
Weight	static	kgs	*private*
Fat	behavioural	gm	0.08 – 90
Fiber	behavioural	gm	0.06 – 34
Carbs	behavioural	gm	0.06 – 150
Sodium	behavioural	mg	1.92 – 2745
Protein	behavioural	gm	0.14 –75
Calories_burned	behavioural	kcal	1025 – 4331
Resting_heart_rate	behavioural	bpm	49 – 83
Lightly_active_minutes	behavioural	mins	2 – 481
Moderately_active_minutes	behavioural	mins	0 – 211
Very_active_minutes	behavioural	mins	0 – 253
Sedentary_minutes	behavioural	mins	254 – 999
Steps	behavioural	-	162 – 32871

4.2.2 Data Transformation

For preparing the data for training, we first remove the *Date* and *Gender* information. Next, the remaining features are normalized and fed to the model for training. Depending on the selected privacy setting (noisy input), we can provide DP-input data to the GAN, which will enable the generation of DP-synthetic samples, as stipulated by the post-processing theorem. Since the data contains categorical or static attributes, which require higher privacy settings, we add Laplacian noise with $\varepsilon = 0.2$ to ensure high noise addition and consequently, stronger privacy settings. On the other hand, behavioral attributes possess a lower risk of re-identification. So we add Laplacian noise with $\varepsilon = 0.5$ to ensure sufficiently high noise addition without losing data utility.

4.2.3 Model Training

As mentioned earlier in the Section 2.2, we use BGAN for synthetic data sample generation. We trained the BGAN by sampling the population based on gender and geographical location. Moreover, we trained the model in three different privacy settings (non-DP, noisy input, and noisy output) as will be briefly explained in Sec. 4.4.

4.2.4 Data Inverse-transformation

Once the training is complete and the data is generated by the generator, we first de-normalize the features to better reflect the original data ranges. Afterwards, the columns *Date* and *Gender* are appended to the generated data to make a complete record. Moreover, depending on the chosen privacy setting (noisy output), we add Laplacian noise with $\varepsilon = 0.2$ to the static and with $\varepsilon = 0.5$ to the behavioral variables.

4.3 Proposed GAN Network for Synthetic Smart Health Care Data Generation

4.3.1 Generator Network

The generator network is shown in Fig. 4.2. The network takes an input of 15×1 signal followed by 2 dense layers with 64 and 32 neurons, appended with a Leaky ReLU activation with a rate of 0.2. The last dense layer acts as the output layer which takes *tanh* as an activation function.

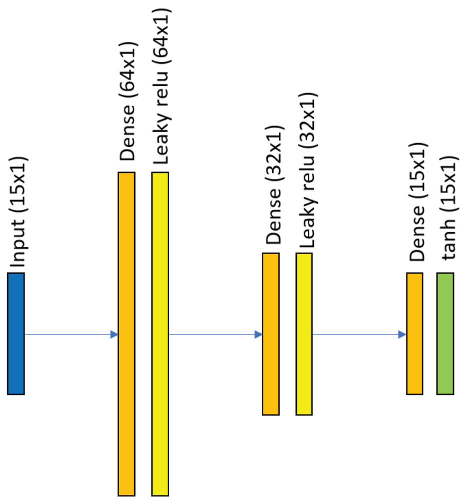


Figure 4.2: Generator network.

4.3.2 Discriminator Network

The discriminator network is shown in Fig. 4.3. The network takes an input of 15×1 signal followed by 2 dense layers with 512 and 256 neurons. Both the layers take Leaky ReLU as an activation function with a rate of 0.2. The last layer of the Discriminator network is a dense layer with 1 output and applies *sigmoid* as an activation function.

4.3.3 Learning rule

We use adaptive moment estimation (Adam) optimizer for both the Discriminator Network and the final GAN network, which computes the adaptive learning rate for each network weight over the learning process from estimates of first and second moments of the gradients. For the configuration parameters, the learning rate α is set to 0.0002 and the exponential decay rate for the first β_1 and second β_2 moment estimates are set to 0.5 and 0.999 respectively. The epsilon that counters the divide by zero problem is set to $1e-8$.

4.4 Experiments, Results and Discussion

We now present our experiments with different additional privacy settings, the respectively generated samples and their histogram distributions.

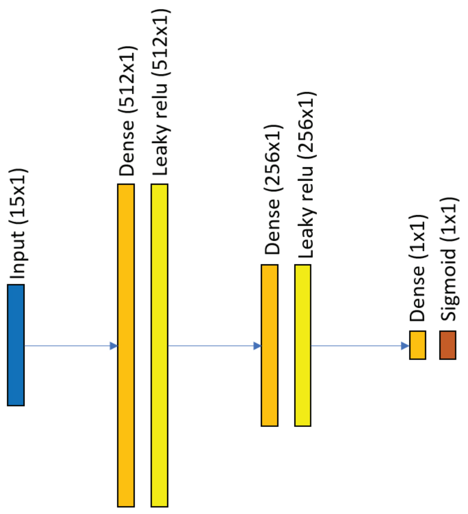


Figure 4.3: Discriminator network.

4.4.1 Experiments

As shown in Fig. 4.1, our pipeline offers multiple points for Laplacian noise addition for differential privacy, enabling 3 experimental settings. The GAN network is able to learn the distribution and to produce plausible examples in each case.

4.4.1.1 Synthetic Data Generation with no Noise Addition

In this setup, the original data is taken as input by the GAN network and the aim is to generate plausible results close to the original data (non-DP).

4.4.1.2 Synthetic Differentially Private Data Generation

In this setup, the network is trained on differentially private data i.e., DP is applied prior to sending it to the discriminator (noisy input). This allows the GAN model to generate differentially private synthetic samples. This approach may be beneficial for settings where synthetic data generation is offloaded to a third party, or when the threat model includes the server node.

4.4.1.3 Applying Differential Privacy to the Synthetic Data

In this setup, we apply DP to the generated synthetic data to observe the effect of noise addition on the quality of generated data (noisy output). This approach offers the advantage in terms of control on the noise addition in generated data, depending on the sensitivity of the data features. However, it requires the additional computation of noise that is added to each generated data point individually.

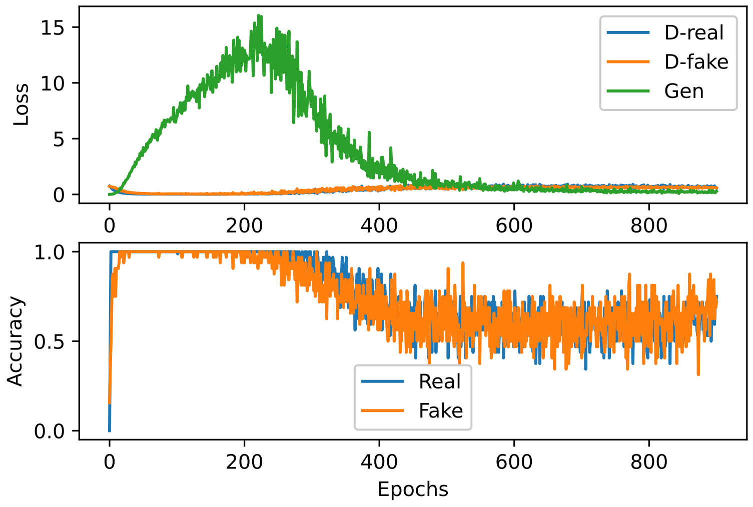


Figure 4.4: Line Plots of Loss and Accuracy for a Stable GAN.

Table 4.2: Example data samples from Belgium population. `Age` and `Gender` are hidden.

Dataset	Height	Weight	Fat	Fiber	Carbs	Sodium	Protein	Calories Burned	Resting HR	Active Minutes				Steps
										Lightly	Moderately	Very	Sedentary	
Original	169	66.18	39.0	11.0	33.0	1189.0	4.0	2308.08	59.526	121	6	28	731	9706
	169	66.18	39.0	7.0	125.0	1125.0	34.0	2707.35	61.99	166	13	56	732	14070
	169	66.18	33.0	8.0	96.0	1361.0	66.0	2485.35	58.45	99	22	60	774	12008
BGAN	175	87.09	29.20	7.80	73.46	1192.83	41.48	2556.28	63.44	157	63	78	768	12222
	175	87.09	41.32	10.71	49.00	1072.10	33.07	2873.35	64.92	195	49	84	772	14374
	175	87.09	60.82	11.69	98.62	1447.21	31.67	3286.82	64.77	253	56	73	889	14877
BGAN w/ DP output	177	93.62	30.51	13.55	70.19	1191.99	42.48	2555.69	66.29	154	60	71	772	12222
	177	93.62	37.94	9.63	50.15	1071.65	34.7	2875.86	70.71	195	48	81	772	14374
	177	93.62	62.06	9.84	94.18	1447.51	32.17	3286.16	70.32	252	49	73	892	14875
Original DP	161	66.78	39.04	10.02	33.97	1195.83	5.42	2308.34	57.32	121	8	26	732	9704
	161	66.78	38.06	7.31	125.42	1122.46	32.66	2706.42	67.77	164	9	56	732	14074
	161	66.78	29.79	2.82	99.02	1360.55	65.02	2485.37	57.75	97	23	62	777	12005
BGAN w/ DP input	181	80.74	33.14	4.12	99.12	1020.8	39.5	3099.14	58.69	213	51	23	757	9769
	181	80.74	22.25	11.02	38.29	1416.57	4.838	2611.83	58.09	137	2	3	754	9944
	181	80.74	58.99	11.19	104.60	483.94	41.96	2593.24	59.56	190	48	106	732	12004

4.4.2 Results and Discussion

Our proposed approach generates synthetic and private smart health care data using BGAN in combination with DP. The GAN network is stable, and able to

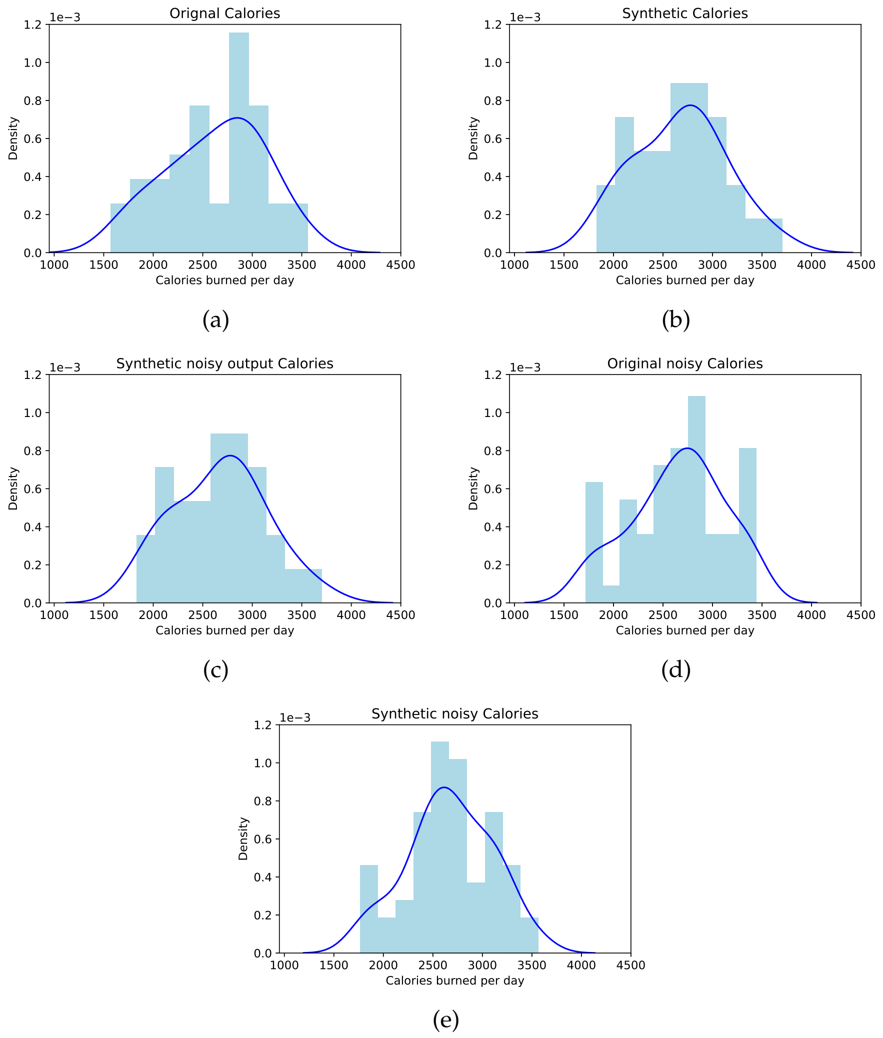


Figure 4.5: Histogram distributions for calories burned per day (kcal). Samples: Belgium males with `RHR=70-75bpm`.

generate plausible results. Figure 4.4 shows the stability of the proposed GAN where the top subplot shows line plots for the discriminator loss for real samples (blue), the discriminator loss for generated fake samples (orange), and the generator loss for generated fake samples (green). It can be seen that the three losses are somewhat unstable early in the run before stabilizing around epoch 420 to epoch 600. Losses remain stable after that, showing the stable behavior of the GAN, although the variance increases. The discriminator loss for real samples and fake samples is around 0.5, and loss for the generator is slightly higher between 0.5 and 1.0. It is expected the model will generate plausible samples between epochs 420 to 600. The bottom subplot shows a line plot of the discriminator accuracy on real (blue) and fake (orange) samples during training. Similar behavior can be seen as seen in the subplot of loss i.e., the accuracy starts off quite different between the two sample types, then stabilizes between epochs 420 to 600 at around 60% to 70%, and remains stable beyond that, although with increased variance.

Table 4.2 shows an example of few rows from the real dataset, and the synthetic rows were generated by the trained GANs. Here, *Original* and *GAN* represent datasets samples with no noise addition (non-DP). *GAN with DP output* shows generated data samples with DP-noise addition (noisy output). Similarly, *Original DP* and *GAN with DP input* represent the original DP-input and the generated synthetic samples respectively (noisy input). As can be seen, all the generated examples look realistic in all the selected privacy settings.

In order to see if the generated data and the real data both come from the same distribution, we show a visualization of the respective histograms. As an example, we only consider the distribution of the burned calories from the logs of male participants belonging to Belgium. It can be seen from Fig. 4.5 that the original (Fig. 4.5 (a)) and the synthesized (Fig. 4.5 (b)) burned calories follow more or less the same kind of distribution indicating the good performance of the proposed BGAN network. We also perform the Kolmogorov–Smirnov (KS) goodness of fit test [144] on the samples taken from original and synthetic calories distributions, which gives a *p-value* of 0.98, indicating a high probability that these samples are from the same distribution and showing that the proposed BGAN is indeed able to learn the diverse categorical and numerical features and generates realistic synthetic samples.

The distribution of original DP data samples (noisy input) is depicted in Fig. 4.5(d), which exhibits a similar distribution as the DP data generated by BGAN shown in Fig. 4.5(e). Moreover, the KS test on original noisy calories distribution (DP input) and the synthetically generated noisy calories distribution gives a *p-value* of 0.97, indicating a high probability that the samples come from the same distribution and the proposed BGAN is able to generate differentially private sample distributions.

Using DP input allows us to generate differentially private data instead of explicitly applying DP to all the synthesized data samples. On the other hand, applying DP-mechanism after synthetic data generation allows for more control in terms of noise addition, and consequently, data utility. As can be seen in Fig. 4.5(c), the distribution of the samples is retained although the records are noised and differentially private.

All our experiments and results demonstrate that although the proposed GAN architecture is quite simple, yet it achieves very high performance in terms of generating both synthetic and differentially private synthetic data. The Fitbit-based smart health care dataset possesses highly diverse features and the proposed DP-mechanism with BGAN is stable and generates high utility synthetic data.

4.5 Summary

We have proposed a system for creating synthetic and private smart health care datasets using BGANs and differential privacy. Using a real-world collection of Fitbit-based smart health care datasets, we tested our proposed approach in three privacy preservation settings. Our proposed approach is able to learn categorical and numerical values for highly diverse tabular data distributions, and we obtain stable GANs trained for dataset generation. As a result, we generate realistic synthetic smart health care datasets that possess similar (and enriched) distributions as the real data while preserving user privacy. Our proposed method for smart health care data generation also allows control for different privacy settings and paves way for the publication of open smart health care datasets for sharing and use in research and industry.

Our proposed approach has some limitations that arise from usage of GANs for data generation. Just like many ML methods, GANs need a huge amount of data to produce useful results. For example, we need a large amount of samples for each category of users (based on gender and location) to generate diverse examples of new users and to not have a bias towards a particular user (generating more examples that resemble a particular user). Moreover, GANs require constant optimization of the generator and discriminator because if the network is not tweaked properly, it will result in producing data similar to each other.

Acknowledgment

We are truly grateful to the anonymous participants who volunteered to provide their data for this study. I would like to thank Muhammad Arsalan from Otto-von-Guericke-Universität Magdeburg, Germany, for his support, constant feedback and the pair programming sessions. I would also like to thank Sonia-Florina Horchidan from KTH for helping with the data imputation.

CHAPTER

5

Machine Learning with Reconfigurable Privacy on Resource-Limited Computing Devices

5.1 Introduction

With the increasing popularity of the Internet of Things (IoT), a tremendous amount of data is continuously being acquired by a variety of intelligent sensor nodes. This data is then processed over central cloud platforms or is transformed through some edge devices before reaching the central processing node [145]. Although this strategy enables data storage management and big data processing while utilizing a distributed computing paradigm, however, distributed processing and storage also lead to data integrity and privacy concerns. Firstly, cloud-based platforms may misuse data due to monetary goals such as product marketing. In such cases, the privacy contracts are designed underhandedly to conceal such privacy breaches [146]. Secondly, cloud-based platforms undergo security and privacy breaching attacks from time to time [147], [148]. Thirdly, the end-user might not be comfortable sharing his private information or publicly disclose some of his data. In a nutshell, although cloud-based distributed platforms enable big data processing and storage with certain guarantees of the quality of service, however, privacy preservation of such data is vital from the user privacy perspective [149], [11].

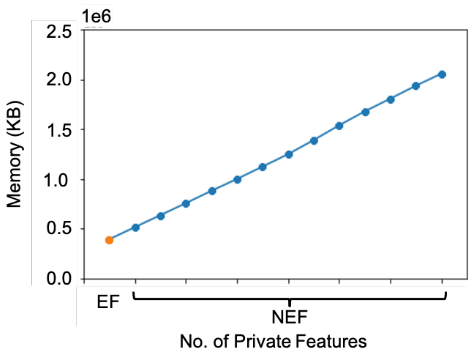
Privacy should be preserved on the data acquisition site i.e. mobile device or embedded edge device attached to some sensors nodes (SN) before the data is transmitted to the central cloud-computing resource [150]. In all cases, mobile devices or embedded edge devices have limited battery, memory and processing resources. Moreover, the way devices utilize the available bandwidth effectively, also known as spectrum efficiency, is a critical performance metric in 5G communication with a large number of devices [151]. Therefore, privacy preservation in resource constraint data acquisition devices becomes a bottleneck in data processing pipelines.

The SN acquires a particular set of data features depending on the application or usage scenario. For example, in the case of a fire alert system, the features would be temperature, humidity, and presence of smoke. In the best-case scenario, the system would like to preserve the privacy of all possible features. However, preserving some features may not be essential or does not influence the privacy of the user even if they are not preserved, such features are called non-essential features (NEF). On the other hand, the application might have some features whose privacy must be preserved at all costs, known as the essential features (EF). In the optimal case, the embedded devices should preserve the privacy of EF as well as the NEF if the resources such as storage, bandwidth, and processing capability of the devices permit. In order to adequately utilize the system resources, the crucial question of which subset of NEF must have privacy preservation should be addressed to best utilize the storage, bandwidth, processing and memory resource of the embedded edge device without significantly compromising the accuracy of the algorithm.

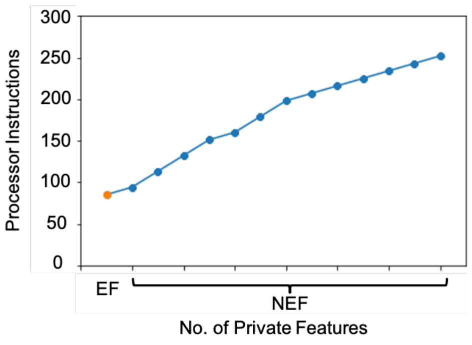
Another important issue concerning the machine learning (ML) applications is the application accuracy, since privacy may affect the performance of the training and utility of the ML system model [5, 152, 153]. Therefore, the accuracy of the resulting ML training model is a significant constraint while selecting the optimal subset of NEF.

One way to design such a system involves application-specific feature selection by performing different trials or by using application design experience. The handpicked feature selection may work in some application scenarios, however, they are challenging to design, are unscalable and unadaptable. Instead of having handpicked feature selection, a scalable approach would be to dynamically choose NEFs adequately suited to the constrained resources of the devices. To avoid rewriting privacy encoders for varying scenarios, a microservice architecture may provide privacy as a service approach.

Although it might be tempting to apply privacy preservation measures on all the input data features to ensure maximum provision of privacy, privacy comes at the cost of increased resources and very often, a negative impact on accuracy and efficiency of the system [5,152,153]. Figure 5.1 shows the trend of increasingly private features with device resource constraints for an edge device, Raspberry Pi model A for a simple machine learning application using anonymization techniques for privacy preservation (more details to be presented in Section 5.2.4). As can be seen, adding privacy preservation requires a significant increase in resource consumption. Moreover, depending on the privacy preserving technique employed, the system could use up to twice the resources with a significant drop in efficiency and accuracy of the system. For example, cryptography-based privacy preserving solutions could consume up to 2-5 seconds per operation [6,154], which is undesirable for IoT-driven



(a) Memory consumption



(b) Processor Instructions

Figure 5.1: Increasingly private features vs. resource consumption for an ML-based application with data anonymization.

systems.

On the other hand, using low levels of privacy may not only violate the rights of users, expose the system to privacy-breaching attacks, but also may violate the data protection laws such as EU's General Data Protection Regulation (GDPR). Therefore, it is important to find optimal operating conditions offering a good trade-off between system performance and the level of privacy preserved. Moreover, in the case of resource constrained devices, it becomes vital to employ efficient privacy preserving practices on selected features to ensure the best functionality and quality of service. In summary, the privacy preservation must be done dynamically in a way that the privacy of all EFs is preserved and the most optimal set of NEFs is selected constrained to memory consumption, processing requirements, bandwidth consumption and accuracy of the algorithm. Moreover, the additional processing time required for privacy preservation should not cause violation of the service-level agreement between the service provider and the clients.

The contributions of this work are as follows.

- We propose and present a novel approach with corresponding algorithms and an end-to-end system pipeline for reconfigurable[1] data privacy in machine learning on resource-limited computing devices.
- We present and have developed a novel greedy search algorithm, DIGS, to find the optimal selection of privacy-preserved input data features provided device resource constraints for a given machine learning algorithm with its input and output data features.

[1]Here we use the words *reconfigurable* and *tunable* data privacy as synonyms.

- We propose an end-to-end system pipeline with our proposed DIGS algorithm, as well as injective privacy preservation functions using generalization anonymity techniques for reconfigurable privacy.
- We have implemented, illustrated and evaluated the results of our proposed approach using a real-world smart health care dataset and machine learning application on a resource-constrained edge device.

Evaluation of our proposed approach for reconfigurable privacy in machine learning on resource-limited devices shows significant resource savings, with up to 26.21% memory, and 16.67% CPU instructions, and 30.5% network bandwidth savings as compared to making all input data features private.

The rest of the chapter is organized as follows. The proposed technique and the DIGS algorithm are presented in Section 5.2. Our experiments and results are presented in Section 5.3 and discussed in Section 5.4. Finally, we present related work in Section 5.5 followed by summary of conclusions and future work in Section 5.6.

5.2 Proposed Technique for Machine Learning with Reconfigurable Privacy Preservation

Given an ML application with its set of input data features (both EF and NEF), we first calculate the cost for the NEF and pass these costs to the optimization algorithm. The algorithm creates a cost matrix using the input and selects the most optimal features within the range of device resource constraints. We then apply privacy preservation to the selected NEF additionally with the EF. This new privacy encoded dataset is used for the ML application. The complete system pipeline is shown in Figure 5.2. We now explain the system model followed by the explanation for each component of the proposed system pipeline.

5.2.1 System Model

Consider a system of P producers and A consumers (applications and services), where P is set of N producers and A is set of M consumers respectively, as shown in formulas below.

$$\mathbf{P} = \{P_1, P_2, ..., P_N\}$$

$$\mathbf{A} = \{A_1, A_2, ..., A_M\}$$

Each P consists of a certain number of features associated with the producer, for each i^{th} producer the set Σ_i is shown in formula below. To preserve the privacy of

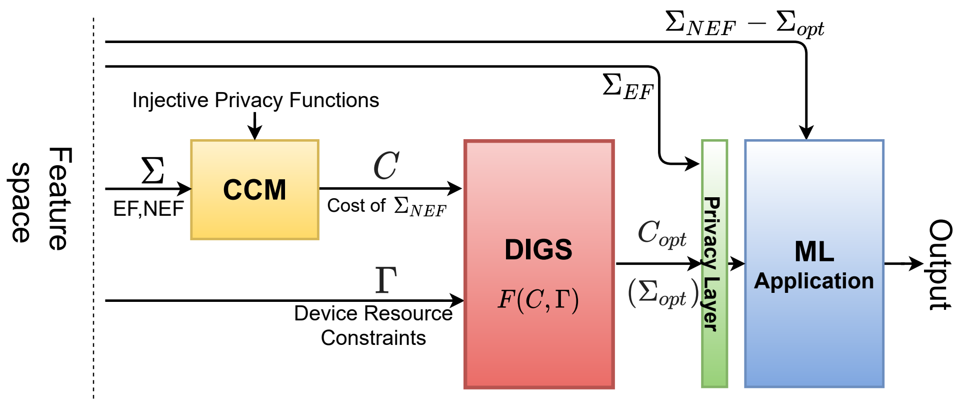


Figure 5.2: System pipeline with DIGS.

the user, the σ's of each P_i are encoded in a certain format before they reach the consumer, as shown in the following formula.

$$\Sigma_i = P_{i\sigma} = \{i\sigma_1, i\sigma_2, ..., i\sigma_x\}$$

where x is the total number of features and $i\sigma_1$ represents the feature 1 produced by i^{th} producer and so on. Now, these features are collected at the edge node such that the feature set becomes:

$$\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_x\}$$

where σ_1 represents all the instances of feature 1 collected from N producers and so on.

However, encoding all Σ is not efficient in terms of certain constraints Γ such as: memory constraints Γ_m, bandwidth constraints Γ_{bw}, processing constraints Γ_p (number of instructions or operations), prediction accuracy constraints Γ_a, and storage constraints Γ_s. In simple words, Γ contains the maximum threshold for all these constraints:

$$\Gamma = \{\Gamma_m, \Gamma_{bw}, \Gamma_p, \Gamma_a, \Gamma_s\} \tag{5.1}$$

Therefore, the subset of Σ (Σ_{opt}) is selected to meet the given Γ using a certain optimization function $F(\Sigma, \Gamma)$, where $\Sigma_{opt} \subset \Sigma$. We use set notation to represent the features and constraints because there can only be one occurrence of each feature and its respective constraint, and the initial order of the set denotes the feature numbers and constraint type which help define the order of the cost matrix in the following Section. The complete producer-consumer system model is depicted in Figure 5.3, where the users are shown on the left hand side.

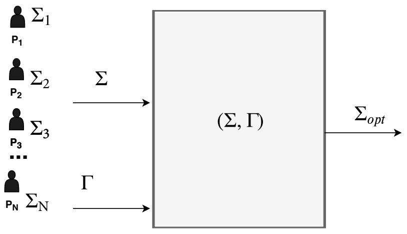


Figure 5.3: Producer-consumer system model.

The major goal of $F(\Sigma, \Gamma)$ is to find a function which maps Σ to Σ_{opt} constrained by Γ. The function $F(\Sigma, \Gamma)$ is tunable in the sense that each time the constraints are updated, new Σ_{opt} can be generated.

5.2.2 Greedy Optimization Algorithm - DIGS

We present a greedy approach to selecting a set of optimal features for provided system constraints, *Dynamic Iterative Greedy Search* (DIGS) for privacy preservation.

Assumptions: We make the following assumptions for this algorithm. The features are assumed to be independent as the feature similarity or correlation is not catered to in the algorithm. Moreover, the service provider should specify the essentially private features as *EF* and the optional features as *NEF* as this is orthogonal to the functionality of the DIGS algorithm.

$$\text{EF} \cap \text{NEF} = \emptyset$$

DIGS: Consider that C represents the cost matrix of all constraints for each feature in Σ, as shown in the following formula.

$$C = \{C_m, C_{bw}, C_a, C_s, C_p\} \tag{5.2}$$

Where, each element in C corresponds to the type of constraint (such as C_m) which in turn has x elements, the σ corresponding to the total number of features. For example, the memory cost constraints can be represented as:

$$C_m = \{C_{m\sigma1}, C_{m\sigma2}, ..., C_{m\sigma_x}\}$$

Moreover, the cost of making a feature private on the embedded edge device is calculated collectively for all the producers. For a set of features to be optimal, all its subsets should be optimal. This implies that the optimality has a property of monotonicity. In order to select the optimal features for the provided device resource constraints, DIGS performs a breadth-wise iterative search on the NEF. That is, we first check if the cost of each feature is within the maximum resource consumption allowance and afterwards, we check the combinations of features and their respective costs against the maximum resource consumption allowance.

The DIGS algorithm works as follows. Consider a cost matrix C with the 3 rows corresponding the the constraints C_m, C_{bw}, and C_p. We first represent each element of each row (containing constraints C_m, C_{bw}, C_p) in C as a key-value pair such that *key* represents the feature number (same as column number) and *value* contains the cost of making that feature private. For example,

$$C_m = \{(\sigma_1 : C_{m\sigma 1}), (\sigma_2 : C_{m\sigma 2}), ..., (\sigma_x : C_{m\sigma x})\} \tag{5.3}$$

Then we sort each row in ascending order to minimize the calculation time. Sorting each row in ascending order helps minimize the calculation time as we would like to make as many features private as possible, which means the features that consume less resources should be combined first for Σ_{opt}. As we mentioned earlier, for a set of features to be optimal, all its subsets should be optimal. Therefore, we generate a list of subsets for the set of viable features at each step. Starting from subsets of cardinality 1 in Algorithm 1 assuming all the features as viable (optimal), we rule out all the subsets of features of cardinality 1 that violate any of the resource consumption constraints. This gives us the list of subsets with optimal features of cardinality 1, the opt_feature.

Next, we generate all possible subsets of features from opt_feature growing in cardinality at each step in Algorithm 2, and rule out (block) all the subsets of features that violate any of the resource consumption constraints. In other words, for each category of performance constraints (memory, bandwidth and so on) as represented by Γ, we select the set of features Σ_{opt} represented as C_{opt} that are optimal for a particular constraint (*locally optimal*) as well as optimal for the total constraints (*globally optimal*). The resultant global_optimal contains the list of all possible globally optimal subsets which are then passed on to Algorithm 3. Algorithm 3 selects the subset(s) of the highest cardinality (containing the most number of features) and afterwards, sorts them by their overall resource consumption in ascending order. The subset of features with the highest cardinality having the lowest resource consumption is the most optimal combination of features to make private. A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage. We call the DIGS algorithm greedy because the algorithm makes the optimal choice at each step (with increasing cardinality) as it attempts to find the largest overall optimal subset of features to make private provided the resource consumption constraints.

The current version of DIGS is written in Python. It must be noted that Σ_{opt} contains only the optimal NEF that can be made private under the given device resources constraints for a target ML application.

Algorithm 1 The Greedy Algorithm: Combination of Features of Size 1

```
Function DIGS($C, \Gamma$) : opt_feature
    Represent each element $x$ in the cost matrix $C$ as a key-value pair:
    $x \leftarrow (key : val)$, where $key$ is feature number (same as column number) and
    $val$ is the cost of making that feature private
    $sortedC \leftarrow$ sort elements of each row in $C$ by $val$ in ascending order so that
    features with the lowest costs appear first
    $subset \leftarrow$ generate a list of subsets of keys of cardinality 1 in $sortedC$
    Initialize empty lists opt_feature and blck_feature
    foreach $y$ in subset do
        $suby = y$
        foreach $i$ in $\Gamma$ do
            forall key in blck_feature do
                if $key \subseteq suby$ then
                    blck_feature.append(suby)
                    break
            end
            foreach $j$ in suby do
                $sum \leftarrow$ the cost of making feature $j$ private for given constraint $i$
                if $sum > \Gamma[i]$ then
                    blck_feature.append(suby)
                    break
            end
        end
        opt_feature.append(suby)
    end
    return opt_feature
end
```

5.2.3 Calculating the feature costs - CCM

An important supporting component of DIGS is the cost calculation module, **CCM**. The programming logic consisting of the application code as well as the programming language, and the Σ (containing both EF and NEF) are provided as an input to the CCM. The presented version of CCM currently supports Python programming language and can be easily extended to support other programming languages.

5.2.3.1 Memory consumption

We calculated the memory consumption of each feature in the dataframe after applying the injective privacy function and we also calculated the memory consumed while running the respective injective privacy function. For the dataframe memory,

Algorithm 2 The Greedy Algorithm: Additional Features

```
Function DIGS_add_feature(sortedC, Γ, opt_feature) :(add_opt_feature,
global_optimal)
    Initialize add_opt_feature, add_blk_feature and global_optimal
    foreach x in range(2, len(opt_feature) + 1) do
        subset ← list of subsets of opt_feature of size x
        Initialize gt_sum and gt_cons to 0
        foreach y in subset do
            suby = y
            foreach i in Γ do
                forall key in add_blck_feature do
                    if key ⊆ suby then
                        add_blck_feature.append(suby)
                        break
                end
                Initialize sum to 0
                foreach j in suby do
                    sum ← add to the previous sum, the cost of making feature j
                    private for given constraint i
                end
                if sum > Γ[i] then
                    add_blck_feature.append(suby)
                    break
                else
                    gt_sum += sum
                    gt_cons +=Γ[i]
                end
            end
            if suby is not in add_blck_feature then
                add_opt_feature += suby
                global_optimal.append([suby, gt_sum, gt_cons])
        end
    end
    return add_opt_feature, global_optimal
end
```

Algorithm 3 The Greedy Algorithm: Most Optimal Features

```
Function most_optimal(global_optimal) : most_opt_set
    Select the sets of features with the most number of elements from
    global_optimal
    most_opt_set ← Sort the sets by their overall resource consumption in
    ascending order and select the set that has the least total resource consumption
    return most_opt_set
end
```

we simply used Python's `memory_usage()` function and derived the memory for all the input data features. For calculating the memory consumed by the functions, we used a Python library called `guppy()` [155] which provides the current heap memory. First, we cleared the heap memory with `setrelheap()`. Then we executed the injective privacy functions with different input data features and printed the heap memory again. We took the difference and added this memory usage (in kilobytes) to the dataframe memory to obtain the overall memory consumption.

5.2.3.2 Bandwidth consumption

The bandwidth cost is calculated by dividing the dataframe memory with the network speed of the selected edge device in a certain network, such as 4G. It is calculated in kbps.

5.2.3.3 Computational processing cost

The computational cost is device-specific. We calculated the computational processing cost by translating injective privacy functions into processor instructions via the lookup table of the selected edge device. For each injective privacy function, we tokenize the code to identify instances of operations such as `if, else, and, or` statements. These statements are translated to number of processor instructions using the lookup table from the device instruction set.

All the computed costs are normalized depending on the type of resource constraint. This allows DIGS to make fair comparisons between the resource consumption costs for each data feature for different type of resource constraints.

5.2.4 SVM Classification Model - ML application

Another important component of our system pipeline is the target ML application. For this work, we use a supervised classification model using SVM in order to evaluate the effect of our privacy preserved data on the application accuracy and resource consumption. Our dataframe contains these features: nutritional value intake (calories, fat, fiber, protein, carbs, sodium), activity data (lightly active minutes, sedentary minutes, very active minutes, moderately active minutes, calories burned, steps count), heart rate, height, weight, age, and gender. We created custom “labels" based on the data values in the features. We followed the guidelines from [156] and [157] to label each day of the user as healthy or unhealthy. For example, on a specific day if a user was consuming more than 2000 calories and not being active, and if the user’s BMI is over the average range, we labeled this day as being unhealthy (1), otherwise healthy (0). We used standard scaler in order to have similar distribution of the dataset as we noticed there were more records with one label than the other. We ran the model on different versions of data, such as all non-private data, all privacy encoded data, dataset with only the EF being private, and the dataset containing DIGS selected private features and essential private features. We compare the accuracy of these datasets to see the effect of added privacy on the performance of ML application.

5.3 Experiments and Results

5.3.1 Dataset

For the evaluation of our proposed system, we used two datasets: a first-hand collected Fitbit dataset and GAN-augmented Fitbit dataset. Table 5.1 shows an overview of the datasets in terms of scale. The dataset collection and processing are described in next Sections.

Table 5.1: Datasets used for evaluation

	Fitbit	**FitBit-GAN**
# of Users	25	500
# of Days	60	60
# of raw Records	≈17 M	≈340 M
Size	3.2 GB	≈64 GB

5.3.1.1 Fitbit dataset collection and privacy concerns

We use the first-hand collected Fitbit dataset as described in Section 4.2.1 using *Fitbit Charge 2 HR* devices with 25 subjects with some additional data processing steps as follows. The missing entries for nutritional breakdown of consumed meals we imputed using Nutritionix API [143]. The body-mass-index (BMI) was calculated from the user weight and height using the BMI formula [158]:

$$\mathbf{BMI} = \mathrm{kg}/\mathrm{m}^2 \tag{5.4}$$

Moreover, the calorie intake is calculated by converting the macronutrients (grams) in into calories as:

$$\textbf{Calories\_in} = fat \times 9 + protein \times 4 + carbs \times 4 + fiber \times 1.5 \tag{5.5}$$

These conversions are done according to [159] and recommendations by Food and Drug Administration (FDA), USA.

The naming convention used for the features is the same as feature names imported from the Fitbit API, and other features like height, weight, gender and BMI are user-defined. The records are aggregated per day and the complete spectrum of the dataset features is presented in Table 5.2. Here, the less frequently updated variables are termed as static variables.

Table 5.2: Dataset

Feature Name	**Type**	**Description**
Date	Static	Data log date
Age	Static	Age of the user
Gender	Static	Male or female
Height	Static	Height of the user
Weight	Static	Weight of the user
Fat	Behavioral	Fat (gm) consumed from each food
Fiber	Behavioral	Fiber (gm) consumed from each food
Carbs	Behavioral	Carbohydrates (gm) consumed from each food
Sodium	Behavioral	Sodium (mg) consumed from each food
Protein	Behavioral	Protein (gm) consumed from each food
Calories_in	Behavioral	Calculated calorie intake
Calories_burned	Behavioral	Calories burnt
Resting_heart_rate	Behavioral	Resting heart rate on the day before Date
Lightly_active_minutes	Behavioral	Minutes of light activity
Moderately_active_minutes	Behavioral	Minutes of moderate activity
Very_active_minutes	Behavioral	Minutes of high activity
Sedentary_minutes	Behavioral	Minutes spent sedentary
Steps	Behavioral	Steps taken
BMI	Static	The BMI of user
labels	Target	Indicates whether the user diet and activity on Date is healthy/unhealthy

On one side, the amount of sensitive information is quite huge making the active collection of data hard as it requires fully informed consent for data disclosure. On the other side, experimenting on such dataset demonstrates the capabilities of privacy preservation techniques because of the highly private information present in the dataset. However, we removed the sensitive individually identifying data (using anonymization) for the actual processing.

5.3.1.2 GAN for Synthetic Data Generation

We used Bi-directional GANs [160] with no noise addition settings (similar to the network shown in Section 4.3) to extend the collected dataset. The data generation was done separately on the 4 different dataset splits based on location and gender. As shown in Table 5.1, we generated 60 days of data for 500 users, which amounted to 33120 aggregated daily records in total.

5.3.2 Device Resource Constraints

We employed Raspberry pi RPi 1 Model A for evaluating the DIGS algorithm. RPi is installed as an edge device which collects data from multiple sensors. RPi has 256 MiB of memory; however, on average, approximately 56 MiB is used by the device itself for its system operations. Besides system memory usage, typically, around 40% of the memory is used by other applications and processes. Therefore, effective available and usable RPi memory is around 120 MiB. Consider a scenario where 100 devices are connected to the edge device, which is a reasonable sensor node size assumption. Our experimentation has revealed that encoding of EF with the load of a hundred sensors consumes almost 20% of the memory. In our experiments, four EF, i.e., weight, height, age, and gender, are always considered private. In a nutshell, the device has limited operational memory left, and the algorithm has to decide the selection of NEF. The situation becomes further complicated with more complex injective privacy functions or more sensors. Similar constraints can be observed for the device's processing capabilities; for the available 700 MHz, mostly 20% of the available computing power is typically used for system operations. Furthermore, in our observations, on an average 40% is utilized for other processes of the machines. So effectively, around 250 MHz clocking power is available for 100 devices. We have employed simple injective privacy functions during the experiments, which worked easily for EF with available device resources. Increasing the complexity of the injective privacy functions revealed that computational resources are much more relaxed than memory in terms of feature encoding and selection. Finally, the bandwidth required to transmit private EF over the network for all devices was accommodated in 40% of the available bandwidth. However, several unnecessary features might cause congestion over a network, or in the case of other traffic bandwidth, over 40% of the bandwidth might not be available. Nevertheless, in our experiments, DIGS could dynamically negotiate required NEF for available bandwidth, memory, and computational power.

5.3.3 Privacy Encoder Functions

The proposed pipeline uses user-defined injective privacy encoder functions to preserve the privacy of processed user features. For our experiments with both the

Fitbit and Fitbit-GAN datasets, we used injective privacy encoder functions employing generalization-based anonymity techniques. As mentioned in Section 3.3.1.1, generalization involves replacement of specific values with more general ones for an attribute [7]. For example, the numeric feature (or attribute) is generalized by partitioning the attribute domain into intervals, similar to a binning approach.

EF. Since we assume that the edge device always has sufficient resources for making the EF private, the injective privacy encoder functions are always applied to the EF before passing them to the ML application. For example, *Age* is assumed to be an EF for our experiments and its respective privacy encoder function is as follows (applied to individual data records, as an injective function):

- if $Age < 25$, replace the age record with `Young`
- else if $25 <= Age < 30$, replace the age record with `Below_30`
- else, replace the age record with `Above_30`

NEF. Each NEF has one respective injective privacy encoder function for our experiments. However, these functions may or may not be eventually applied in the Privacy Layer depending on the NEF selection done by the DIGS algorithm. For example, *Resting_heart_rate* is considered to be an NEF for our experiments, and its respective (injective) privacy encoder function is as follows:

- if $Resting\_heart\_rate >= 100$, replace with `High`
- else if $60 <= Resting\_heart\_rate < 100$, replace with `Normal`
- else, replace with `Good`

Similar generalization transformations are applied to all the NEF. For example, the caloric intake and active minutes are generalized as `Low/High`, and the macronutrients intake is generalized as `Low/Normal/High`. Afterwards, we calculate the resource consumption for making an NEF private by measuring the total resource consumption for transforming all the data records through that NEF's injective privacy encoder function. The following Section presents our results for the Fitbit dataset.

5.3.4 Results for Fitbit Dataset

We first calculate the cost measures such as memory consumption, bandwidth requirements and the processor instructions on the edge device for the non-private version of the Fitbit dataset. The total memory consumed by the non-private data was 247.0 KB. Due to small size of the dataset, the bandwidth required for this dataset on a 100 Mbps 4G network was 0.0198 Mbps. The additional processor instructions required for this dataset were 0 since the dataset is not private.

Afterwards, we transformed every feature from this dataset to private feature using the privacy encoder injective functions customized for each feature. We calculated the memory required for processing of each feature through relevant injective privacy functions. The total memory consumed through this privacy preservation process for all features was 533.378 KB. After converting the features to privacy encoded features the total private dataframe memory was 1750.369 KB. So if we made all the features of the Fitbit dataset private the total additional memory required is 2283.747 KB. The bandwidth required to transfer this data on a 100 Mbps LAN network would be 0.1827 Mbps. The total processor instructions required to convert the non-private features to private feature was 252 instructions in Raspberry pi RPi 1 Model A.

Due to the sensitive nature of our data there are four features that we decided to make private to protect our users, denoted as EF. These EF are: *Age*, *Gender*, *Height* and *Weight*, which must be private regardless of resource constraints. We then calculated the resources required by the dataset that contained these four EF as private. In this case we only converted these four EF through their own injective privacy encoder functions, and all other features were kept with their original values. The total memory consumption through the injective function for these features was 103.395 KB. The memory consumed by the dataframe after enforcing privacy only on the EF was 528.727 KB. After adding both memory usage through the injective privacy encoder functions and the dataframe itself, we got a total of 632.122 KB of memory consumption. The bandwidth requirement would be then 0.0506 Mbps and the processor instructions required to make only the EF private was 85.

In order to provide additional data privacy, now we want to select more features to be private apart from the user-defined EF. Moreover, we would like to not exceed the maximum resources available to us while encoding the data to preserve user privacy. The available resources for our application on the Raspberry Pi device was 2516.584 KB of memory, after deducting the required memory for making the essential features private we had 1884.462 KB of memory available so we can select additional NEF to be private within our allocated memory. We had enough bandwidth and processor to allocate the processing of making our full dataframe private. We applied DIGS to select the additional features that could be private by using the available resources we have in terms of memory, bandwidth and processor instructions. For Fitbit dataset, DIGS selected a set of combination with 9 NEF features that could be private in addition to the 4 private EF. There were 715 different combinations of features from the 13 NEF. After calculating the total resources required by all the 715 combinations, DIGS selected the most optimal set in terms of the number of features and resource consumption. With these additional

9 NEF we can now encode a total of 13 features including the EF and provide more privacy.

After converting the DIGS selected NEF to private features including the EF, the total memory required by the 13 private features through the injective privacy encoder functions was 409.241 KB. The memory consumed by the dataframe after the privacy encoding was 1340.657. Adding the two required memories, we got a total of 1749.898 KB of memory required to make the DIGS selected features private. The bandwidth requirement was 0.1400 Mbps and the total number of additional processor instructions required were 216.

We name these four different versions of the dataset as:

- Version 1: Non-private data
- Version 2: Only Σ_{EF} private
- Version 3: DIGS selected $\Sigma_{\mathrm{opt,EF}}$ private
- Version 4: All $\Sigma_{\mathrm{EF,NEF}}$ private

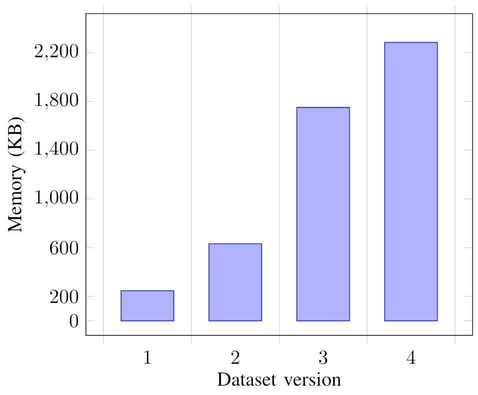
Table 5.3: Additional resources required for different versions of the Fitbit dataset

Dataset	Memory (KB)	Bandwidth (Mbps)	Processor (Instructions)
V1	247.0	0.0198	0
V2	632.122	0.0506	85
V3	1749.898	0.1400	216
V4	2283.747	0.1827	252

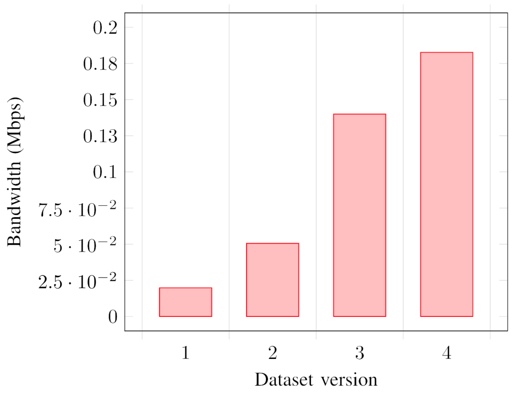
Figure 5.4a displays the increase of memory requirement in KB with the increase of more features being private, Figure 5.4b and 5.4c represents the bandwidth requirement in Mbps and the processor requirements in terms of processor instructions respectively.

We assumed total available memory in the Raspberry pi device after all other factors considered was 2516.584 KB, the available bandwidth was 40000 kbps and available processor instructions were 4.2M. DIGS had a great impact on memory, as the device's available memory was smaller. However, we noticed that the privacy encoding process did not have a notable impact on network bandwidth and processor instructions. The selected bandwidth speed was very high for the required bandwidth speed to convert the features to privacy encoded features, as well the processing power of the RPi device was also a lot higher than what was required to run all the injective privacy encoder functions. Hence the major resource saved by DIGS here was in terms of memory.

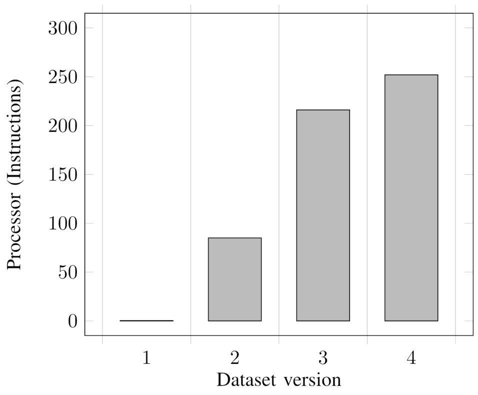
To validate the effect of our privacy preservation method, we ran our ML application on all four versions of the Fitbit dataset. We use one-hot encoding



(a) Memory



(b) Network Bandwidth



(c) Processor Instructions

Figure 5.4: Additional resource consumption for increasingly private versions of Fitbit dataset.

mechanism for representing all the private versions (V2, V3 and V4) of the dataset. We observe that the classification accuracy on each dataset was very close, hence privacy encoding did not adversely affect the application accuracy. Moreover, for the all private versions of data, we gained a higher accuracy as more variables became categorical which favoured the SVM learning mechanism.

Table 5.4 shows the prediction accuracy of different SVM models trained using the four different versions of dataset, the number of private features on each version, the additional memory requirements in order to make the features private, the resources saved when we select the DIGS selected features to be private, and the privacy compromised in terms of number of exposed NEF. For each dataset version, SVM was trained using a train/test split of 80/20 with 10-fold cross validation sets.

We can see that DIGS effectively selected 9 NEF to be private which provides more privacy and also can save 26.21% of memory, 30.5% of network bandwidth, and 16.67% of CPU instructions as compared to making all the features private and overutilizing the resources.

Table 5.4: The result of SVM on increasingly private versions of Fitbit dataset, number of private features, percentage of resources saved and privacy compromised features

Dataset	**Accuracy on SVM**	**Number of private features**	**Additional memory required (KB)**	**Resources saved (%)**	**Number of compromised private features**
V1	96.41%	None	0	100%	All
V2	96.35%	4	385.122	81%	13
V3	96.79%	13	1502.898	26.21%	4
V4	99.62%	All	2036.747	0%	None

5.3.5 Results for GAN Dataset

We discuss the results for GAN dataset in this section. The data processing steps for Fitbit-GAN dataset are the same as that of Fitbit dataset. This dataset is quite large than the Fitbit dataset as it contains 500 users with 33120 aggregated records. The non-private version of the dataset required total 5299.2 KB of memory, the bandwidth required was 0.423 Mbps and the processor instructions required was 0 as we did not run any injective functions on the non-private dataset. We calculated the additional resources required for this dataset in order to make only the EF private and also to make all the features private. The total memory required for the EF being private only was 11169.24 KB. We ran DIGS with each individual feature's resource cost and noticed that as our current edge device has a low capacity for memory as this large dataset could not be accommodated with the required resource allocation in terms of memory. We still ran our ML application to classify the user's day as "healthy" or "unhealthy" in order to compare the result on this dataset with the original dataset. The accuracy of the SVM on the complete non-private dataset was 98.07% and the accuracy on all private dataset was 98.54%.

In order to verify the scalability of our selected edge device's capacity of memory allocation we ran our experiments on three more different sample sizes of the Fitbit-GAN dataset. The first dataset we selected was half the size of the GAN dataset. The dataset now contained 16560 rows of records selected from the large GAN dataset. This dataset required 2649.6 KB of memory for the non-private version of the dataset and the available memory we have for our application was

2516.584 KB. So this dataset was also large for the edge device and we did not continue further with our experiments for the half sized of the GAN dataset.

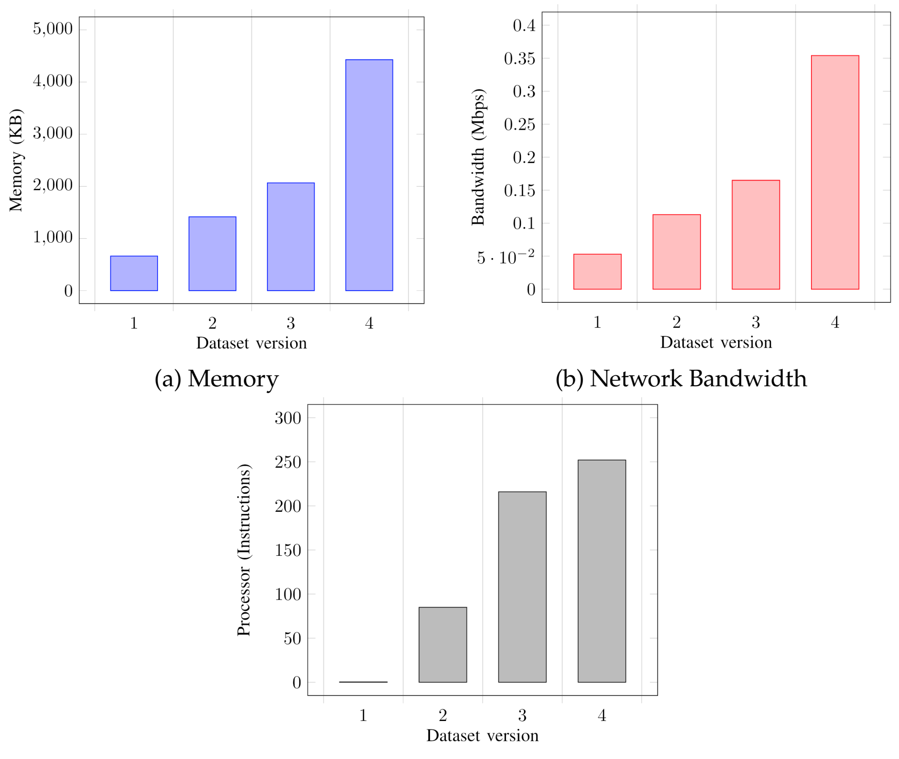
We further downsized to one-quarter of the GAN dataset and the dataset now contained 8280 rows of records that were selected from the large GAN dataset. The memory required for processing this dataset also exceeded the device resource constraints, so we further scaled down to processing one-eighth of the Fitbit-GAN dataset on the edge device.

As can be seen in Figure 5.5, for the one-eighth size of the GAN dataset, the non-private version required 662.24 KB of memory. This dataset contained 4140 rows of records and was a lot smaller than the original GAN dataset. The additional memory requirement for the injective functions to make the four EF private was 1416.19 KB. Moreover, the required bandwidth for the non-private dataset was 0.0529 Mbps and the additional bandwidth for only the private EF was 0.113. The processor instructions required for the non-private and essential features being private was 0 and 85 respectively. So now we have 1100.394 KB to allocate the memory requirement for additional features to be private by DIGS. We ran DIGS to select additional features to be private within our allocated resources. DIGS selected three additional features to be private, *Fat*, *Carbs* and *Protein* among the thirteen additional NEF. These three features consumed the lowest resources altogether. We present all the different resource requirements for different versions of this dataset in Table 5.5.

Table 5.5: Resources required for different version of one-eighth sized GAN dataset

Dataset	**Memory (KB)**	**Bandwidth (Mbps)**	**Processor (Instructions)**
V1	662.24	0.053	0
V2	1416.19	0.113	85
V3	2065.832	0.165	216
V4	4428.896	0.354	252

We tested the different GAN dataset versions as we did for the original Fitbit data by running the SVM and the results are displayed in Table 5.6. For each dataset version, SVM was trained using a train/test split of 80/20 with 10-fold cross validation sets. We notice the trade-off between providing more privacy and resource consumption. If we want to make more private features then we have to allocate more resources such memory in our experiments. If we only make the EF private, then we are compromising privacy for thirteen NEF which would be undesireable as well. By applying DIGS, we were able to select an optimal combination of features to be made private within our available resources while saving extra cost for resources. We were able to save 62.74% memory compared to making all the features private. Also the accuracy on the private versions of



(a) Memory

(b) Network Bandwidth

(c) Processor Instructions

Figure 5.5: Additional resource consumption for increasingly private versions of Fitbit GAN dataset.

the essential features of the data was higher than the non-private version, and the more private DIGS selected private features were a little better than the only EF being private and we got the highest accuracy on the all-private features dataset. One possible reason could be the nature of the supervised classification model combined with the use of one hot encoding for the categorical values, as the SVM only takes numbers as training and test data. The more privacy we enforced, the more categorical values were created for each feature which give better accuracy for more private data.

5.4 Discussion

Our proposed algorithm provides a solution for reconfigurable privacy to make more data features private along with the private EF in order to provide maximum

Table 5.6: The result of SVM on different dataset for one-eighth GAN dataset, number of private features, percentage of resources saved and privacy compromised features

Dataset	Accuracy on SVM	Number of private features	Additional memory required (KB)	Resources saved (%)	Number of compromised private features
V1	96.83%	None	0	100%	All
V2	97.6%	4	753.95	79.98%	13
V3	97.65%	7	1403.592	62.74%	10
V4	98.80%	All	3766.656	0%	None

privacy to the user. The more privacy, the better it is for any kind of personal data, as data can contain very sensitive information. DIGS performs very well with the real life users' data collected from Fitbit dataset. It has successfully selected 9 NEF to make private in addition to the 4 EF. Out of the 17 features that we have excluding the target feature, "*labels*", we are only compromising privacy for 4 NEF. Moreover, by using the optimal combination of private features selected through DIGS, we can save 26.21% memory, 16.67% processor instructions and 30.5% network bandwidth in comparison with the resources required by all the features being private. In terms of the impact of privacy preservation on ML application accuracy, we noticed that the accuracy of our SVM classification has improved while applying more privacy to the data. This SVM model can help our user to check if the user had a healthy day or unhealthy day and can asses the nutrients value that they consumed and the physical activities that they had done on that specific day to plan a better or healthier lifestyle. Our results prove that our ML application provides more user data privacy while respecting available device resource consumption constraints. Moreover, we observe higher prediction accuracy of the SVM model for the all-private data due to the usage of one-hot encoding on all the input data features.

In order to check the scalability of our application, we experimented with the GAN dataset which contained 500 users and 33120 rows of records. As this dataset was very large as compared to our edge device, Raspberry Pi's available resources were not able to process the privacy preservation for this dataset. The non-private dataset itself was large enough not to fit in the device. Then we decided to test on different sizes of this dataset to check the limit of the device's capacity.

We divided the GAN dataset to its one-eighth size and this dataset was small enough to fit within the resource constraints. However, it was still more than 2$\times$ the size of our original Fitbit dataset as it contained 4140 records and the Fitbit dataset had 1663 records. For this dataset, DIGS was able to select 3 additional NEF features to be private in addition to the 4 EF which provided more privacy to the user data. We also could save 62.74% of memory if we compare with all the features being private, even though we are compromising privacy for 10 NEF. But compared to the non-private dataset and only EF being private dataset, with the help of DIGS we were able to make more features private while using the available resources.

The accuracy of the SVM had a similar trend as on our original Fitbit dataset. The SVM trained on the most private dataset version exhibited the highest accuracy due to the one-hot encoding of all the features as they all were categorical after privacy encoding and accuracy gradually improved with more privacy. However in an ideal situation we want all our data to be private and have the highest level of privacy if sufficient device resources were available.

The reason that DIGS could not be applied on a large dataset like Fitbit-GAN was not due to the ML application, but rather it was device specific. We have selected a low-memory edge device and it was not able to handle large memory consumption required for processing the Fitbit-GAN dataset. For edge devices with more resources, our algorithm would work fine as it worked on the two smaller dataset Fitbit dataset and the one-eighth size of the GAN dataset.

5.5 Related Work

Fitness trackers are gaining popularity as they help to maintain user's health and well-being. But in order to provide a fully personalized user experience, these trackers require the users to share their personal information. Sharing personal information becomes a huge concern for the users, as they need to decide whether the tracking devices are a safe platform to share sensitive personal information or not [161]. Data privacy in similar applications and devices has become the biggest concern due to the pervasive nature of huge data collection through different IoT devices and the lack of data security [162]. Various data privacy techniques have been applied to address these concerns and researchers are continuously working on inventing new techniques to provide better protection for the user data throughout the data mining process [161].

Commonly used data protection techniques in the health care sector include k-anonymity [7], l-diversity [8] and t-closeness [9]. The k-anonymity method involves arranging specific columns of quasi-identifiers that are altered or removed, resulting in k rows in the dataset with similar attributes [7]. l-diversity [8] and t-closeness [9] are extensions of the same concept with stronger privacy guarantees. The required modifications are implemented before publishing the data to tackle privacy threats. This is also known as privacy-preserving data publishing. These privacy-preserving techniques work well in general, however, with the increased learning abilities of artificial intelligence-based algorithms, these data protection methods are not enough to reduce various privacy-breaching attacks and threats, as noted in [11, 163].

In the study [21, 164] by Horchidan et al., differential privacy [4] was applied to add noise to the dataset in order to provide privacy to a similar Fitbit dataset. This technique was effective when applied on a large dataset, whereas on a smaller dataset it might produce incorrect results [21,164]. Orlosky et al. [165] studied accelerometer and pulse rate of 24 users from Fitbit Blaze devices. Their study showed that the accuracy on the Fitbit Blaze is not good compared to the medical grade devices and the users' concern about data privacy is valid due to the intervention of third-party applications.

Providing data privacy has some trade-offs. Dong et al. [166] conducted a research to measure the trade-off between smart grid operations and adversarial inferences about consumer conduct, by considering direct load monitoring of thermostatically regulated loads and investigating how its output degrades as it receives less samples. By providing less samples they wanted to protect the privacy of the data. Their work provided a framework to evaluate the trade-off between utility of the collected data and its privacy preservation.

Reconfigurable or tunable privacy provides user control over the trade-off between the user privacy and other factors such as: access to services [167], data sharing to trusted parties in collaborative computing environments [168], system performance [169] and efficiency [170,171], model accuracy [172], and data utility [166,173] and deniability-utility (in case of location-based services) [170].

In our study, we used users' food and activity data and our goal was to make user data as private as possible constrained to device resource consumption. We used generalization techniques in order to provide user data privacy. We also studied the trade-off between provision of privacy preservation and the required additional resource consumption for privacy preservation in a resource constrained environment.

5.6 Summary

We propose DIGS (Dynamic Iterative Greedy Search), a novel mechanism for reconfigurable privacy preservation for ML features on the resource constrained edge devices. DIGS provides reconfigurable privacy by choosing an optimal set of data features to make private provided the device resource constraints. DIGS employs user-defined privacy injective functions to make the data private. We demonstrate DIGS using privacy injective functions employing the anonymization based privacy preservation solutions. Moreover, our privacy preservation mechanism based on user-defined privacy injective functions is flexible as it can cater to any privacy preservation solution as long as each data point is processed individually and the cost for each operation can be computed. Results of our experiments on health care datasets show that for the studied ML application with 17 data features, DIGS is able to select up to 9 additional (non-essential) features apart from the 4 user-defined essential features that must be private and provided additional privacy to the user data, with significant memory savings as well as CPU instructions and network-bandwidth savings as compared to making all the features private. Moreover, the privacy encoded data used in ML applications provides minimal impact on the performance of the model, and in our case, provides even better prediction accuracy due to the use of the one-hot encoding mechanism.

In this chapter, we have implemented and evaluated a proof-of-concept prototype of our proposed mechanism for reconfigurable privacy in ML. Limitations of our work include: the DIGS algorithm is currently tested with scenarios having only one injective privacy encoder function per feature. Moreover, it can only work with privacy preservation techniques that can be represented as injective functions. Our research can be extended in multiple directions as it is a novel approach towards privacy provision. The algorithm can be tested on other edge devices such as micro-controllers like Arduino Uno, smartphones, and also advanced variants of Raspberry Pi that have more resources. We can also extend our system by applying ML-based optimization solutions for selecting the most optimal feature set to be private provided the device resource constraints. Furthermore, we can incorporate other techniques for privacy preservation in the injective functions such as k-anonymity, l-diversity, t-closeness, and even stronger techniques for privacy preservation such as differential privacy. Lastly, we can demonstrate the impact of using our system for other kinds of ML applications performing regression or classification while employing different versions of the same dataset with different user privacy preservation levels.

Acknowledgment

This work is done in collaboration with Zannatun Tania at KTH and Hassan Nazeer Chaudhry at Politecnico di Milano, Italy. I would like to thank Tania for her persistent hard work on the implementation of DIGS and to Hassan for his great ideas, constant feedback and active participation in group programming sessions.

CHAPTER

6

Privacy Preserving Time-Series Forecasting of User Health Data Streams

6.1 Introduction

Advancements in cloud computing and the Internet of Things (IoT) paradigms inspired the creation of highly interconnected heterogeneous computing environments that are capable of generating and processing tremendous volumes of data. Consequently, the past decades witnessed a notable growth in the adoption of wearable smart devices, which further enabled the development of applications and services employing the collected data. For instance, numerous health applications are adopting cloud platforms for real-time health monitoring, achieving fitness goals, disease diagnostics, and medical data analysis leading to personalized medicine. Modern health trackers such as Fitbit are equipped with multiple sensors capable of recording complex health metrics like the caloric burn, sleep quality, or the wearer's activity level [24]. Besides, health-tracking mobile applications, such as MyFitnessPal, are freely accessible to users and can be paired with wearable devices to provide an intricate glimpse into user health [25]. Typically, health and fitness applications require processing enormous amounts of personal user data. In some cases, this data is handled by an untrusted third party for data analytics or machine learning (ML) based services. In light of this discussion, it can be stipulated that personal health information is not absolutely private. Moreover, the General Data Protection Regulation (GDPR) demands to enforce privacy preservation policies, particularly in the health care domain, as this data could be potentially maltreated.

Real-time or low-latency systems and services, such as wearable smart device services, continuously process an enormous influx of data streams. In the traditional health care domain, data anonymization and de-identification are usually employed as the privacy preservation practices. This is owing to their relatively lower complexity, although they lack effective privacy guarantees [11,174]. An alternative approach utilizes synthetic but representative datasets for improved privacy guarantees [124,136]. On the contrary, privacy preservation solutions based on cryptography, blockchains, and private compute units are often compute-intensive, and hence, not suitable for distributed low-latency environments [11]. On the other hand, decentralized privacy-preserving ML-based solutions like federated learning (FL) facilitate collaborative training of models without exposing raw data and can be acclimated to real-time environments [175]. Moreover, solutions based on differential privacy (DP) can be integrated into ML systems to provide strong privacy guarantees. However, these solutions may introduce computational and performance overheads in the system, such as accuracy loss and degraded quality of service. These overheads become critical in time-series forecasting, particularly in the health care domain, as a small error in the forecast may lead to dire consequences. Moreover, devices may experience connectivity issues in distributed environments, so the system necessitates catering to asynchronous learning and service provisioning. All these issues make privacy-preserving forecasting of health data streams a challenging problem. The investigation of privacy preservation trade-off for low-latency environments is a significant research problem, particularly in the context of health data streams, as they demand strong privacy guarantees.

This work proposes a novel technique for privacy preservation on real-time predictions. In particular, an end-to-end pipeline for time-series forecasting is implemented. Furthermore, the impact of applying several privacy preservation solutions on the application performance in terms of prediction accuracy and model training time is studied.

Our contributions can be summarized as follows:

- Design and implementation of an end-to-end pipeline for time-series forecasting of health data streams in a federated learning environment.
- Design and implementation of a clustering mechanism using streaming k-means algorithm and pattern matching.
- Integration of state-of-the-art privacy preservation solutions in the designed pipeline and evaluating their impact on the time-series forecasting.
- Collection and refinement of a real-world dataset from geographically distributed users.
- Creation of a privacy-preserving smart health care dataset employing Generative Adversarial Networks (GANs).

6.2 User Clustering Using Stream Processing

Our end-to-end health forecasting pipeline is implemented in Apache Flink [15]. It has two main components: (1) the clustering mechanism, which includes the streaming k-means subsystem and the pattern matching subsystem, and (2) the FL system with privacy preservation mechanism.

Our proposed pipeline consumes a stream of time-series diet and health logs from several users and attempts to predict the next logs by leveraging the history of each individual and the similarities between users. Due to the volume and velocity of the ingested data, the system has to be highly scalable. Besides, the streaming nature of the problem imposes strict requirements in terms of latency, as the system should be able to provide real-time predictions. And most importantly, the proposed pipeline has to ensure a strict level of privacy preservation because we are dealing with sensitive data.

First, we cluster the users according to their meal logs (by discovering patterns in breakfast, lunch, and dinner) and use this information to build separate federated models for each group. In this way, we ensure that the prediction caters to individuals with unique dietary patterns and lifestyles, thus offering personalised forecasts. We introduce our two-step clustering mechanism, that is able to group multidimensional time-series data based on common characteristics.

Clustering mechanism. We cluster the users' meal logs over a period of 7 days using the streaming k-means clustering algorithm. Fitbit allows users to specify the meal times in the logs as morning snack, breakfast, afternoon snack, lunch, evening snack, and dinner. We group these meal times for each day into 3 major meals so that each user's meal log for a day consists of breakfast, lunch and dinner. We assign the meals in the stream into three groups/clusters that represent breakfast, lunch and dinner.

At the beginning of k-means clustering, first unique meals that appear in the stream are chosen as centroids for each meal group. We have taken three centroids for the k-means clustering to represent our meal groups, i.e. breakfast, lunch and dinner. We chose $k = 3$ after experimenting over a range of k from 2 to 32, and $k = 3$ displayed the best results in terms of managing to cluster the most meals together. Each user's meal is mapped to the closest centroid value during processing. Once the user meals are mapped to clusters for a period of 7 days, pattern matching is done on the clustered data. In pattern matching, the centroid IDs are considered as numbers of sequences. For example, for simplicity, we have a pattern for each user for one day, user 1 has a meal pattern (1,2,1) and user 2 has a meal pattern (1,3,1). Here the numbers indicate the centroid ID of the group/cluster to which the meals (breakfast, lunch, dinner) belonged to. For pattern matching, Hamming distance is computed over the given sequences of centroid IDs or patterns. In the end, we get groups of users with closest meal patterns. These groups are then used to train a separate FL model for each group. It is to be noted that the pattern matching for meals might find a different number of clusters than the already specified k. This clustering approach whilst simple and intuitive, comes with a drawback in terms of implementation, as it has to be performed by a central entity. However, this approach is not only efficient in terms of improvements in training performance but also beneficial in terms of privacy preservation. This is because the central coordinator stores patterns, which are essentially sequences of centroid IDs, and not the raw data, and can not be exploited to extract a specific user from the system.

We now overview the privacy preserving techniques used in the proposed time-series forecasting pipeline.

6.3 Privacy Preservation Techniques in Time-series Forecasting

Since stream processing requires low latency and real-time response, we select the techniques that ensure strong privacy guarantees with low performance overhead in terms of model training time and with a minimal loss in prediction accuracy.

6.3.1 Categorical Data Anonymization

As defined in [176], Individually Identifiable Data (IID) is "data that identifies the person that the data is about, or that can be used to identify that individual". This may contain uniquely identifiable data such as identification number of any kind, be it social, cultural or economical. The GDPR and other related regulatory requirements for privacy generally apply (only) to this IID. We study the impact of sensitive Individually Identifiable Data (IID) on user clustering by using k-modes for categorical data clustering [177] and k-means for numerical data clustering. We performed an initial clustering of users with the private categorical data to inspect its effect on the quality of clustering. k-modes was used as the expert clustering mechanism to validate the correctness of our streaming k-means approach. We observed that our proposed approach can cluster similar users using non-private data. As this work strongly advocates user privacy, we removed all the IID from users' data except gender and location (country). The latter were retained only for the synthetic data generation with GANs, as both attributes have a major impact on diet patterns; and to observe the correctness of the clustering patterns. However, our proposed pipeline does not make use of any IID in any mechanism.

6.3.2 Differential Privacy for Federated Learning

TensorFlow-Federated (TFF) is used for our implementation [178, 179]. We employ two well-established mechanisms for DP: Gaussian and Laplacian mechanisms [175]. We use the Laplacian mechanism for adding noise to user data - *noisy data* - and Gaussian mechanism from TensorFlow privacy library for adding noise to learned gradients for federated learning - *noisy learning*.

Noisy learning. Gaussian noise addition to the output of a function f of sensitivity S_f on database D is defined by:

$$M(D) \triangleq f(D) + N(0, S_f^2 \sigma^2), \tag{6.1}$$

where $N(0, S_f^2\sigma^2)$ is the normal distribution with mean 0 and standard deviation $S_f\sigma$ [175].

TensorFlow Privacy mainly uses a differentially private version of stochastic gradient descent (DP-SGD) to modify the learned gradients. Models trained with DP-SGD provide provable DP guarantees for their input data. It uses two additional hyperparameters with the stochastic gradient descent optimizer: the `clip` and the `noise_multiplier`. The former is used to clip each gradient computed on each training point in a mini-batch. Then, random noise from a Gaussian distribution is sampled and added to the clipped gradients to make it statistically impossible to know whether or not a particular data point was included in the training dataset. A differentially private query `DPQuery` is responsible for clipping gradients computed by the optimizer, accumulating them, and returning their noisy average to the optimizer [180].

Noisy data. For this approach, we use the Laplacian differential privacy by adding noise directly to the aggregated data records. Traditionally, for Laplace mechanism, random noise is drawn from a Laplacian distribution with mean 0 and variance S_f/ε to achieve ε-differential privacy [4]. In this work, all the data points in an aggregated data record are individually noised as we pick random noise samples for each point from a $Lap(0, 1/\varepsilon)$ distribution.

6.4 Proposed End-to-end Private Learning Pipeline

This section presents the proposed pipeline and describes the interaction of each subsystem in the noisy learning and noisy data approaches for privacy preservation.

6.4.1 System Overview

First, raw data points are aggregated as individual user data records. As explained in the background Section 2, the central coordinator uses updates from the clients to improve a global model. The global model is described using a set of hyperparameters. We use Adam optimizer for both client and server, and Standard Federated Averaging algorithm [39] as the aggregation method. Moreover, a random sample of 10% users is used in each round of FL as this fraction achieves a good trade-off between model convergence and computational efficiency [39].

The records are forwarded to the clustering mechanism, where we use streaming k-means algorithm with pattern matching to find similar users as explained in Section 6.2. Afterwards, the central coordinator in FL stores these cluster patterns, which are essentially the sequences of centroid IDs, and not the aggregated raw data records.

Based on the clustered patterns, the server maintains k federated models, where k is the number of groups. Figure 6.1 depicts the process in one communication round for a randomly sampled client. $\mathtt{client}_p$ sends its updates (step 1). Then, the coordinator looks up $\mathtt{client}_p$ and finds that this client belongs to cluster $k-1$ (step 2). The coordinator finds the model that corresponds to cluster $k-1$ ($\mathtt{model}_{k-1}$) and updates it using the information received from $\mathtt{client}_p$ (step 3). In the final step, the coordinator shares the updated model with the client.

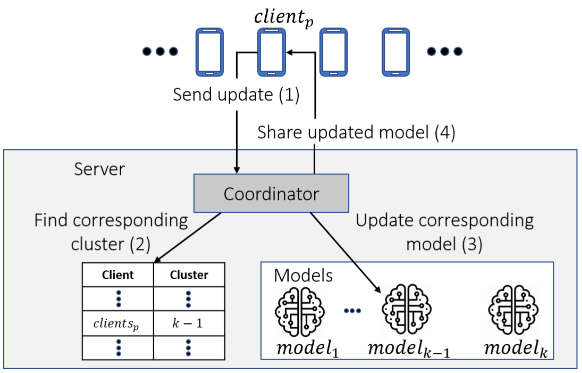


Figure 6.1: An overview of the communication round in the FL process with clustering, assuming grouping into k clusters.

6.4.2 Client Confidentiality with DP

We now explain the proposed pipeline in noisy learning and noisy data settings. Both approaches achieve the goal of noising the updates that are sent to the coordinator so that no sensitive information can be leaked by exchanging these updates.

Noisy learning. The weights are noised and sent to the federated aggregator with the mechanisms provided in TensorFlow Privacy. As depicted in Figure 6.2, each client adds noise to the true update, in both baseline and clustered scenarios and the amount of added noise is controlled by the *clip* and *noise multiplier* parameters, as explained in Section 6.3.2. The standard deviation of the added noise is computed by multiplying these two parameters. Moreover, aggregated user records are also sent separately to the clustering mechanism, as can be seen in Figure 6.2.

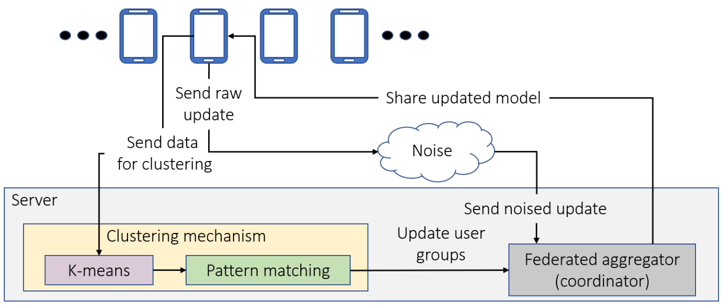


Figure 6.2: Noisy learning: clustered FL using streaming k-means. Baseline model is traditional FL setup (not shown separately).

Noisy data. In this setup, the FL process remains very similar to the standard process, with the small modification that the clients noise their local datasets before trying to improve the federated model. Thus, the weights of each model sent as updates to the coordinator implicitly contain noise. Figures 6.3 and 6.4 show a high level image of the process. It should also be noted that the clustering mechanism receives noised aggregated data records, meaning that the quality of clustering will be affected. It should be noted that the prediction is performed locally and on clean data.

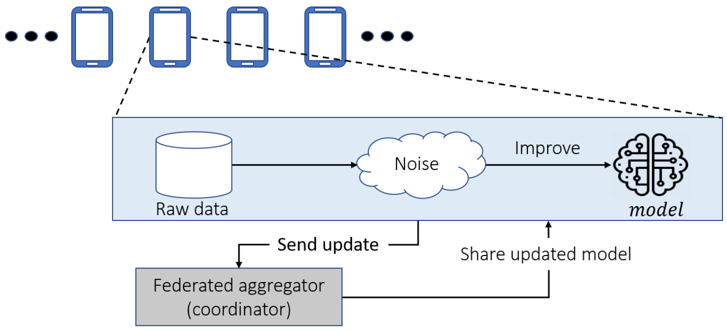


Figure 6.3: Noisy data: a) Baseline FL.

Researchers state that an epsilon higher than 1 does not give good privacy protection in general. However, Apple MacOS's DP has an epsilon as large as 6 and Google's version of DP claims to achieve an epsilon value of 2 in certain scenarios [181]. We will use these values as a guideline in our study and strive to

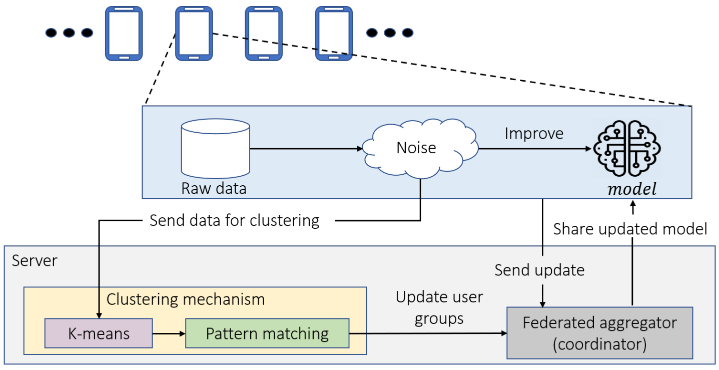


Figure 6.4: Noisy data: b) Clustered FL using streaming k-means.

achieve a compromise between the achieved level of privacy and the performance of the system.

6.5 Experiments and Results

This section presents the impact of introducing our clustering mechanism and DP techniques on the system performance in terms of model training time and prediction accuracy. In terms of privacy, we study the impact of introducing differential privacy as noisy data and learning. Since categorical data anonymization is a one-time process with negligible performance overhead, our measurements do not include the impact of anonymization. Moreover, all the measurements are taken in the FL settings, so the impact of using FL for privacy is not separately measured. We first explain the datasets used in our experiments.

6.5.1 Datasets

We used 3 datasets for our evaluation: MyFitnessPal [26], collected Fitbit dataset and Fitbit-GAN dataset. An overview of the datasets in terms of scale is shown in Table 6.1.

Table 6.1: Datasets used for evaluation

Dataset	# of users	# of days	# of raw records	Size
MyFitnessPal	9.9K	207	1.9M	2.1GB
Fitbit	25	60	≈17M	3.2GB
Fitbit-GAN	630	60	≈435M	≈ 83GB

Table 6.2: Recorded features for Fitbit dataset

Features		Unit	Granularity		Range
			Recorded	Aggregated	
Macro-nutrients	Fat	gm	food	meal	0.03 – 25
	Carbs	gm	food	meal	0.01 – 105
	Protein	gm	food	meal	0.04 – 55
Calories	Burned	kcal	food	meal	416 – 1435
Heart rate	Resting	bpm	7.5s	day	49.5 – 83.4
Activity	Light	mins	day	day	2 – 481
	Moderate	mins	day	day	0 – 211
	High	mins	day	day	0 – 253
	Sedentary	mins	day	day	600 – 998

MyFitnessPal. MyFitnessPal [25] contains records from 9900 users who logged their foods for almost 207 days. Each entry contains an anonymized user ID, logging date, name of the food, and respective macronutrients' breakdown: carbohydrates, protein, and fat. The dataset was analyzed to drop the users with missing or inconsistent logs. Only the users who logged at least one meal per day over a consistent period of time are included. Afterwards, all the meals were aggregated into 3 categories: breakfast, lunch, and dinner. The snacks were summed up with their corresponding meal category (for example, the morning snacks were added to breakfast). After preprocessing, the dataset contained 89 users who concurrently recorded their meals for 151 days. This dataset is used as the best case scenario (no missing values) for our multivariate time-series forecasting experiments.

Fitbit. As described in Section 4.2.1, this dataset collected with Fitbit Charge 2 HR devices had 25 subjects distributed across Belgium and Sweden, with additional data processing steps as follows. Users logs and measurements were exported from the Fitbit platform and Nutritionix API [143] was used to impute the missing nutritional breakdown for meals. The recorded measurements were aggregated into 3 records per day for each user. Each aggregated record contained the nutritional breakdown for a meal (breakfast/lunch/dinner), calories burned during the mealtime, resting HR from the previous day, and activity records for that day. The complete spectrum of data ranges is shown in Table 6.2. It must be noted here that sedentary minutes do not include sleep time, estimated with the recorded sleep activity. As can be seen, the users exhibit all kinds of natural behaviour, ranging from very sedentary to highly active users.

As mentioned in the earlier Chapters, a huge amount of individually identifiable data is collected by the Fitbit platform, therefore the number of participants is relatively small as it requires fully informed consent for data disclosure. However, the biggest advantage of experimenting on this dataset is the private information available for each user. Moreover, as discussed in 6.3.1, all the IID were removed for actual processing.

Fitbit-GAN. We employed conventional Generative Adversarial Network [23] (Section 2.2) to generate augmented Fitbit data for training our models, similar to the network described in Section 4.3 with no noise addition. We refer to this dataset as Fitbit-GAN. A sample of synthetically generated data records for two days for a user is shown in Table 6.3. It can be seen that the GAN is able to learn the macronutrients breakdown for meals, calories burned, resting HR and daily activities.

Table 6.3: Synthetic data generated using GAN

Meal	Fat	Carbs	Protein	Calories Burned	Resting HR	Active Minutes Lightly	Active Minutes Moderately	Active Minutes Very	Active Minutes Sedentary
Breakfast	2.97	35.04	10.56	515.27	64.21	170	22	10	768
Lunch	11.97	47.83	25.64	655.85 8	64.21	170	22	10	768
Dinner	12.73	46.46	21.06	679.38	64.21	170	22	10	768
Breakfast	3.30	28.92	10.76	505.71	64.67	171	23	12	723
Lunch	13.92	49.80	14.40	665.38	64.67	171	23	12	723
Dinner	13.61	54.84	15.91	869.65	64.67	171	23	12	723

6.5.2 Results - Federated Learning without Privacy

In all the experiments, we predict the health data streams for the next day (3 meals' breakdown, calories burned, HR and activities) based on the previous day (3 meals' breakdown, calories burned, resting HR and activities). Moreover, for all the experiments, we train the individual models for each user using 80% of the recorded data streams and use 20% of data streams for testing purposes. We discuss our findings and highlight the contributions they bring in the context of these research questions:

- How accurately can we predict the dietary and health-related behaviour of a user?
- Does grouping similar users bring any benefit in terms of accuracy and/or model training time?
- What impact do privacy preservation methods have on the accuracy of the forecasting?

6.5.2.1 Choosing the right model

We first compared the frequently used statistical models for multivariate time-series forecasting with popular NN architectures to choose the best configuration for this task. For statistical models, we used Vector Autoregression (VAR), Vector Autoregression Moving-Average (VARMA), and Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX). For NN architectures, we experimented with FNNs, LSTMs and GRUs. We used a sample of 10% of the MyFitnessPal dataset to perform our study. Our results showed that neural networks are better candidates when it comes to our use case, as shown in Table 6.4.

Table 6.4: Comparison between the observed error of various statistical models against NN architectures.

	MAE	RMSE
VAR	3.230	4.105
VARMA	7.160	9.051
VARMAX	3.535	4.460
FNN	0.461	0.530
LSTM	2.035	2.621
LSTM-2	1.690	2.151
GRU	1.670	2.102

Next, we performed a grid search for the following hyperparameters using the learning rate ($\eta = 0.001$) for the NN architectures: batch size (b) and the number of neurons on each layer l. Table 6.5 shows the results of the best three models obtained, tested for a different number of epochs e and rounds r in a federated learning (FL) process. We chose the LSTM-2 architecture, i.e. LSTM with two hidden layers with 5 epochs per round for our following experiments as it offers the best prediction accuracy.

Table 6.5: Accuracy of the FL models using a grid search

Best models	**Parameters**	**Federated 40e - 5r**		**Federated 20e - 10r**		**Federated 10e - 20r**		**Federated 5e - 40r**	
		MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
FNN-2	$\eta = 0.001$ $b = 32$ $l = 10$	14.351	17.94	9.238	10.717	4.77	5.525	1.33	1.603
LSTM-2	$\eta = 0.001$ $b = 32$ $l = 200$	11.278	14.398	6.391	7.965	2.35	2.69	0.655	0.853
GRU-2	$\eta = 0.001$ $b = 32$ $l = 200$	10.226	12.703	16.11	18.473	4.104	5.064	1.289	1.621

6.5.2.2 MyFitnessPal dataset

We now present and analyze our results for the MyFitnessPal dataset.

Baseline FL model. We analyze the accuracy of the model trained on the whole dataset and refer to it as the baseline FL model in this case. Table 6.6 shows the achieved performance by training the model with parameters discovered using the grid search in terms of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). As can be noted, our model can forecast the next time-series with an MAE equal to 0.246, meaning that each predicted value will be at most 0.246 gms higher or lower than the ground truth. Given that the macronutrients in this dataset have a minimum value of 0.5 and a maximum of 609, we believe our system is capable of making highly accurate predictions.

Clustered FL models. Next, we analyze the performance of training a separate model for each cluster of users. Our clustering method identified 4 user clusters in this dataset. Table 6.6 shows the prediction accuracy of the FL models. It is important to remember that all the models in this comparison were trained using the same hyperparameters and number of epochs as the baseline FL model. Our results show that most of the clustered models are able to achieve high accuracy, though not as high as the baseline model. In particular, cluster 3 shows the highest increase in observed error. However, cluster 3 contains only 8 users and the least amount of data. So, we believe the data might not be enough for the model to learn from in this case.

Table 6.6: Comparison between the baseline model and the clustered FL models. Dataset: MyFitnessPal.

FL Model		MAE	RMSE	Change in observed error	Training time (sec)
Baseline		0.246	0.319	-	5966
Clustered	**Cluster 1 (14 users)**	0.712	0.811	+2.89×	1082
	Cluster 2 (43 users)	0.536	0.642	+2.18×	3036
	Cluster 3 (8 users)	1.298	1.521	+5.28×	691
	Cluster 4 (10 users)	0.436	0.548	+1.77×	822

We now analyze the training time needed for each model. Even though the clustered models did not outperform the baseline model, it is noteworthy that we are still able to achieve accurate predictions with a drastic decrease in training time. For example, cluster 4 manages to predict the macronutrients with an MAE equal to only 0.436 after training for a period which is around 7× smaller than the baseline model.

Figure 6.5 shows the evolution of the MAE on the training dataset. It can be noted that clusters 1 and 4 seem to learn at the same pace as the baseline model. As expected, the model struggles to learn from the data on cluster 3. As for cluster 2, the model showcases a slower decrease in MAE over the training rounds. However, it should be noted that the evolution of the error is depicted using a logarithmic y-axis.

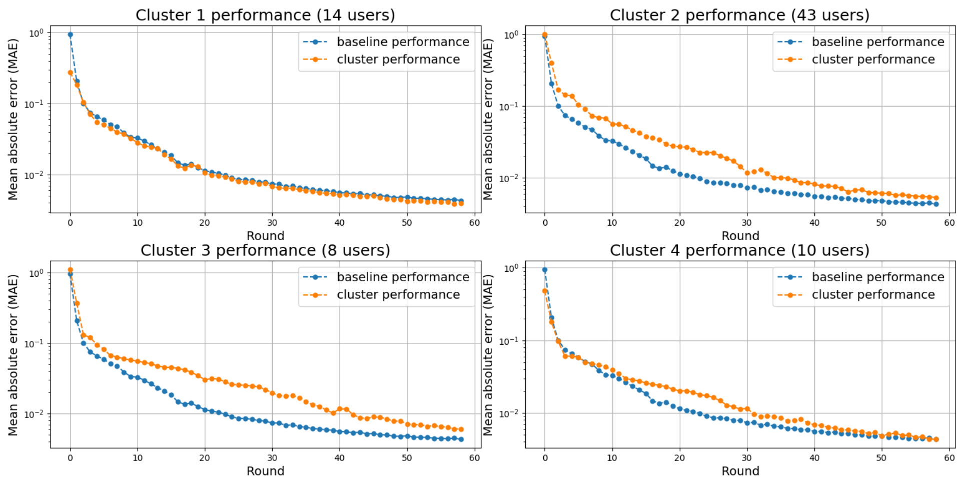


Figure 6.5: The evolution of the Mean Absolute Error of the baseline model against clustered model during the training process on the MyFitnessPal dataset.

Discussion. This set of experiments leads to the following conclusions: (1) we can accurately predict the macro-nutrient breakdown of meals, (2) clustering similar users does not improve the prediction accuracy with small clusters, but (3) we are still able to achieve high performance in terms of significantly less training time.

It is challenging to say why the clustering mechanism failed to improve accuracy. One possible reason could be the small number of users available for each cluster for the clustering mechanism to actually exhibit its benefits. Another plausible cause could be the homogeneity of the dataset. In this case, even if our method managed to discover some groups, the similarity between members of the same group might not be high enough for the model to benefit from. A third cause might reside in the process of choosing an appropriate model for our data. As mentioned before, we ran a grid search to find the parameters that would perform the best on our entire dataset. However, doing so optimizes the model for the entire dataset. Using the same configuration for the clusters might train an overfitted model, which in turn becomes a potential source of performance drop.

6.5.2.3 Fitbit dataset

The Fitbit dataset serves as a real-world example for our specific use-case, as it exhibits gaps in the time-series. This dataset contains 25 users and more private information, such as resting HR and active minutes throughout the day. Similar to the previous dataset, we performed a grid search to find the optimal hyperparameters.

Baseline FL model. We first investigate the performance of our FL procedure on the entire dataset. Table 6.7 shows the accuracy of our baseline FL model. It should be noted that the number of training rounds needed to be adjusted due to the size of the dataset. Training for a longer period led to overfitting. Here we notice an overall accuracy drop as MAE is higher as compared to the MyFitnessPal dataset results. While the MyFitnessPal dataset achieved a prediction with MAE equal to 0.246 (Table 6.6 for the baseline model, the same methodology on the original Fitbit dataset achieves a much higher MAE in prediction accuracy, 3.27. This is probably caused by two factors: a considerably smaller amount of training data and gaps in the time-series. Nonetheless, predicting the next macronutrients intake with a precision of ± 3.27 gms is still remarkable. The same reasoning can be applied to other features.

Table 6.7: Prediction accuracy of the baseline model. Dataset: Fitbit

Predicted	MAE	RMSE	Training time (sec)
Macronutrients	3.27	4.047	245
Calories burned	11.831	15.261	
Resting HR	0.859	1.044	
Active minutes	4.495	5.320	

Clustered FL models. We cluster our users based on their similarities. Our method found 4 clusters of various sizes. We trained a separate FL model for each cluster. Table 6.8 contains the prediction accuracy of these models. The average change in observed error is in comparison to the model trained on the entire dataset. An increase in average observed error suggests a decrease in the model prediction accuracy. Figure 6.6 shows the evolution of the train error throughout the learning process. Three out of four models seem to have a learning curve similar to the baseline model. However, it is obvious that cluster 1 struggles to learn the patterns in its data. On a deeper analysis, we notice that, even thought cluster 1 and cluster 2 have the same number of users, cluster 1 contains the least amount of data, meaning that the logs of the users that belong to cluster 1 are very scarce.

Discussion. As expected, the clusters with the less training data show a considerable decrease in accuracy (clusters 1 and 2). Groups that have more training

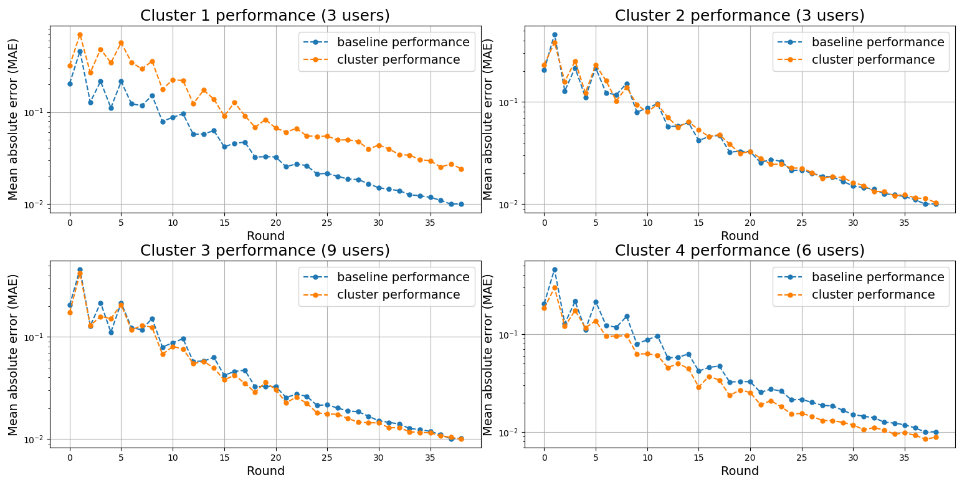


Figure 6.6: The evolution of the Mean Absolute Error of the baseline model against clustered model during the training process on the original Fitbit dataset.

data manage to achieve better predictions. For these reasons, the baseline model outperforms all clustered models in this case. Although this dataset contains very interesting yet sensitive data, we believe that the amount of data in this dataset is a major impediment to drawing relevant conclusions. Hence, we focus on the Fitbit-GAN dataset for further experiments.

6.5.2.4 Fitbit-GAN dataset

The experiments performed on the Fitbit-GAN dataset aim to investigate whether a higher amount of data influences the outcome as compared to our previous experiments. The hyperparameters for the baseline model in this section have been chosen using a grid search. We now evaluate the performance of the FL mechanism.

Baseline FL model. Firstly, we focus on determining how well we can predict the features of our dataset. We perform multi-step forecasting and compute the MAE and RMSE as before. Table 6.9 shows the performance achieved by the model trained on the entire dataset. Similar to the MyFitnessPal dataset results, we conclude that our model can predict user behaviour very accurately, as our prediction is at most 0.025% as far from the ground truth value with respect to the range of each feature.

Clustered FL models. We cluster the users and train separate models for each group. Again, the baseline model outperforms the clustered models for all features.

Table 6.8: Prediction error obtained by training clustered FL models. Dataset: Fitbit

FL Model	Predicted	MAE	RMSE	Average change in observed error	Training time (sec)
Cluster 1 (3 users)	Macronutrients	7.776	9.429	+5.63×	64
	Calories burned	57.544	77.070		
	Resting HR	9.462	12.321		
	Active minutes	19.303	23.290		
Cluster 2 (3 users)	Macronutrients	6.651	7.780	+3.08×	62
	Calories burned	38.816	51.437		
	Resting HR	5.269	6.754		
	Active minutes	12.923	15.459		
Cluster 3 (9 users)	Macronutrients	4.119	4.933	+1.9×	110
	Calories burned	21.791	28.054		
	Resting HR	2.447	3.093		
	Active minutes	7.555	9.120		
Cluster 4 (6 users)	Macronutrients	3.571	4.293	+1.84×	88
	Calories burned	28.165	33.643		
	Resting HR	2.012	2.566		
	Active minutes	7.093	8.612		

Table 6.9: Prediction accuracy of the baseline model. Dataset: Fitbit-GAN

Predicted	MAE	RMSE	Training time (sec)
Macronutrients	0.806	1.054	9891
Calories burned	11.395	14.460	
Resting HR	0.044	0.058	
Active minutes	2.365	2.983	

We note that optimizing the hyperparameters for the entire dataset might lead to overfitting for smaller chunks of the dataset as the data distribution for the groups will be different than the one for the entire dataset. We ran a grid search for optimal parameters for each cluster and found a less complex model configuration to be appropriate for the groups. Moreover, it has been found that the same model configuration can cater to all groups (a small FNN-2 and the same hyperparameters). This will also lead to a major improvement in training time.

Figure 6.7 shows that the clustered models with optimized parameters seem to learn faster and more efficient than the baseline model. It can be noted that, after 100 rounds, each clustered model achieves a smaller training error than the baseline model.

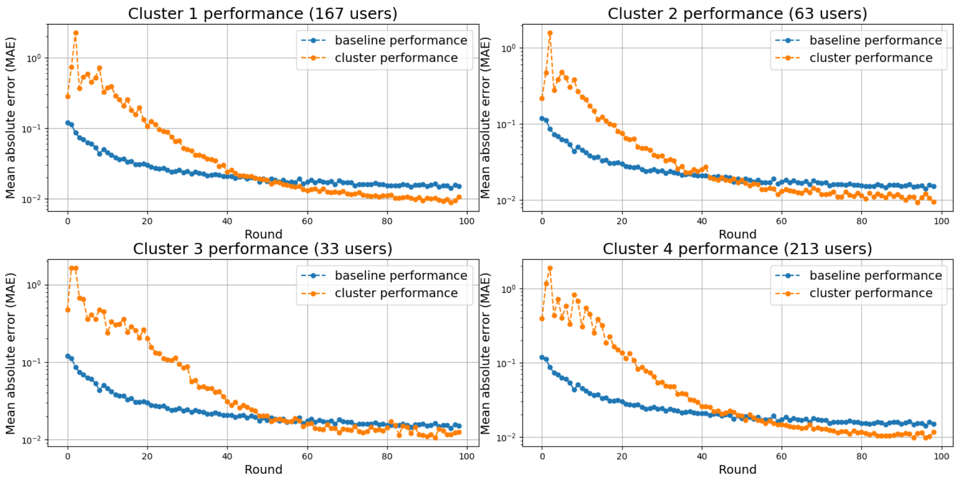


Figure 6.7: The evolution of the Mean Absolute Error of the baseline model against clustered model during the training process on the augmented Fitbit dataset with grid search performed on each group of users.

Table 6.10 shows a significant increase in prediction accuracy for three out of four clusters as compared to the baseline performance. Moreover, the highest training time, which has been recorded for cluster 4, is $20\times$ faster than the training time of the baseline model.

Discussion. In order to address the impact of the cluster size on the overall performance of the system, it should be noted that the clusters discovered by our mechanism are imbalanced. Our results might give the impression that the accuracy is directly proportional to the size of the dataset. However, this assumption is incorrect. The cluster size influences the results only when the NN does not have enough data to learn from. As long as the group contains enough data, the similarity between the members of each group can be leveraged to improve the prediction accuracy. We can, for example, consider the unclustered version to be a considerably larger cluster. The model trained on this group achieves poorer performance than the ones trained on real but smaller clusters. This proves that the similarity between users is the only factor that determines the increase in accuracy, and not the size of the dataset. It should also be noted that clustering improves the prediction accuracy as well as significantly improving the training time.

Table 6.10: Prediction accuracy for training one model per cluster of users. Dataset: Fitbit-GAN

FL Model	Predicted	MAE	RMSE	Average change in observed error	Training time (sec)
Cluster 1 (167 users)	Macronutrients	0.458	0.553	-49%	374
	Calories burned	4.359	5.369		
	Resting HR	0.031	0.037		
	Active minutes	0.829	1.017		
Cluster 2 (63 users)	Macronutrients	0.639	0.788	-17%	199
	Calories burned	6.296	7.792		
	Resting HR	0.04	0.0485		
	Active minutes	2.459	2.822		
Cluster 3 (44 users)	Macronutrients	1.199	1.467	+57%	167
	Calories burned	12.152	14.65		
	Resting HR	0.088	0.107		
	Active minutes	4.101	4.884		
Cluster 4 (213 users)	Macronutrients	0.527	0.63	-47%	478
	Calories burned	5.493	6.8		
	Resting HR	0.02	0.031		
	Active minutes	1.18	1.468		

6.5.3 Results - Differentially Private Federated Learning

The impact of adding privacy preservation methods in the FL mechanism is evaluated by investigating the performance of the system by first noising the learning process and then noising the data itself. We again use a train/test split of 80/20 for all our experiments and predict the health data streams for the next day (3 meals' breakdown, calories burned, HR and activities) based on the previous day (3 meals' breakdown, calories burned, resting HR and activities). The average results for each configuration are obtained by running each experiment at least 3 times.

6.5.3.1 Noisy learning

In this case, noise is added to the updates and sent to the federated aggregator using TensorFlow Privacy mechanisms.

Baseline private FL model. First, a baseline FL model for all users is trained. The results are presented in Table 6.11. Adding Gaussian noise with a higher standard deviation decreases the prediction accuracy of the model but increases the achieved privacy level. Even for the lowest level of privacy, with $\varepsilon = 10.3$

corresponding to the standard deviation (sd) of 0.225, the model performance is approximately 35× poorer than the non-private baseline model.

Table 6.11: Results for noising the learning process to achieve DP in the baseline scenario. clip and noise specify the clip and noise applied to the gradients, with sd standard deviation. sd increases from left to right, suggesting that more noise is added. Contrarily, ε decreases from left to right, suggesting that better privacy levels are achieved as we add more noise.

		clip = 0.3 noise = 0.75 (sd = 0.225)	clip = 0.5 noise = 0.75 (sd = 0.375)	clip = 0.75 noise = 1.2 (sd = 0.9)	clip = 1 noise = 1.3 (sd = 1.3)	clip = 1.5 noise = 2 (sd = 3)
Macronutrients	**MAE**	25.899	24.637	36.308	37.176	77.838
	RMSE	26.835	25.325	37.62	38.457	79.022
Calories burned	**MAE**	678.038	423.555	698.392	818.229	881.288
	RMSE	687.112	437.704	709.221	837.541	900.358
Resting Heart Rate	**MAE**	1.133	2.065	3.089	3.859	4.129
	RMSE	1.208	2.115	3.1581	3.939	4.194
Active minutes	**MAE**	45.942	77.553	104.254	116.546	176.347
	RMSE	49.342	80.0	107.647	121.102	180.743
Epsilon (ε)		**10.3**	**10.3**	**4.27**	**3.8**	**2.2**

Clustered private FL models. When applying noise to the clustered scenario, our results showed that the models became unable to learn anything from the data, as they showcased an MAE higher than 1000 in some cases. Therefore, the results are not presented in this work.

Discussion. The main reasons behind the drastic decrease in prediction accuracy is the number of participants in the federated averaging algorithm which plays a major role on the achieved model performance [182]. Since our dataset is very small as compared to the one used by [182], our results are justifiable. Moreover, the fact that clustering the users caused an even greater performance drop in our experiments is also to be expected in this context, as the number of participants decreases even more when the model is trained for each separate cluster. This analysis shows that the noisy learning method is not appropriate for our use-case.

6.5.3.2 Noisy data

In this scenario, we add noise to the data itself. Each participant in the learning process learns from the noised data and tries to improve the FL model. It should also be noted that, when training on clusters of users, the clustering mechanism also receives noised data, hence, the clusters change for each experiment.

Table 6.12: Results of noising the training data to achieve DP in the baseline scenario.

		$\varepsilon = 2$	$\varepsilon = 1$	$\varepsilon = 0.5$	$\varepsilon = 0.1$	$\varepsilon = 0.025$	$\varepsilon = 0.01$
Macronutrients	**MAE**	0.743	0.715	0.752	0.874	0.984	1.336
	RMSE	0.969	0.943	0.995	1.135	1.378	1.843
Calories burned	**MAE**	11.596	11.937	12.231	14.383	14.891	19.634
	RMSE	14.687	15.451	16.116	18.482	19.903	27.392
Resting Heart Rate	**MAE**	0.048	0.055	0.058	0.067	0.077	0.122
	RMSE	0.065	0.07	0.077	0.088	0.108	0.161
Active minutes	**MAE**	2.483	2.246	2.293	2.344	2.832	3.832
	RMSE	3.012	2.953	2.768	3.159	3.941	5.438
Average increase in observed error		**2%**	**3%**	**7%**	**21%**	**37%**	**94%**

Baseline private FL model. We study the baseline scenario with only one private FL model for all the clients. Table 6.12 shows the effect of various levels of data privacy on the model performance. Google can achieve DP with $\varepsilon = 2$ in certain conditions [181], and this is taken as a starting point for our experiments. Laplacian noise with 0 mean and $\frac{1}{\varepsilon}$ variance is added to the data. With $\varepsilon = 2$, we see an average increase in prediction error of only 2%. Moreover, the best trade-off between privacy and accuracy is obtained for $\varepsilon = 0.1$ where ε is ranging from 1 to 0.025. With this setting, we observe an increase in prediction error of around 21%, with good accuracy and a very high level of privacy is observed.

Clustered private FL models. We now focus on the clustering mechanism and its impact in the FL context. Firstly, as expected, clustering users with noised data alters the clusters. It can be noted that adding more noise to the data has two effects: (1) overall, fewer users are assigned to clusters, and (2) the algorithm discovers a higher number of smaller clusters (the k remains the same but the pattern matching mechanism discovers more clusters than the non-noisy data). As the noise increases, the similarity between users decreases. Hence, the users that were previously very similar might still be clustered together, but the overall size of the groups is expected to decrease. The increased number of found clusters can be observed when applying privacy levels of 0.025, where the pattern matching algorithm finds 5 clusters, instead of 4.

We examine the results obtained with the best ε value we discovered in the baseline case, 0.1. Table 6.13 shows that unlike the noisy learning case, clustering the users can still improve the quality of prediction, even if the data is noised. The clusters show that several users maintain some degree of similarity that can be used to boost the accuracy of the model.

Discussion. Noising the data improves prediction accuracy without any measurable effect on the training time. Adding noise to the data has been popularly

Table 6.13: Results of noising the training data to achieve DP in the clustered FL scenario. $\varepsilon = 0.1$ noise is added. A decrease in the average observed error (compared to baseline model) implies an increase in accuracy.

FL Model	Predicted	MAE	RMSE	Average change in observed error	Training time (sec)
Cluster 1 (73 users)	Macronutrients	0.982	1.131	-20%	222
	Calories burned	9.897	12.379		
	Resting HR	0.039	0.049		
	Active minutes	1.869	2.283		
Cluster 2 (37 users)	Macronutrients	1.025	1.209	-6%	150
	Calories burned	11.96	14.961		
	Resting HR	0.052	0.061		
	Active minutes	2.218	2.68		
Cluster 3 (161 users)	Macronutrients	0.593	0.715	-38%	392
	Calories burned	7.56	9.43		
	Resting HR	0.031	0.038		
	Active minutes	1.842	2.208		
Cluster 4 (97 users)	Macronutrients	0.587	0.712	-34%	262
	Calories burned	9.653	11.463		
	Resting HR	0.033	0.04		
	Active minutes	1.861	2.225		

used as a regularization technique for deep NN to avoid overfitting and improve accuracy. This could explain the very high accuracy we obtain, and a relatively small performance drop compared to the non-noised model. Composing the overall ε experienced at the user end is outside the scope of this work. In principle, removing a user's data has no measurable impact on the overall clusters as well as the prediction accuracy of the system, although more data is shared in the clustering scenario. So, we believe that the overall experienced ε or privacy loss at the user end is also very small as the data is highly noised. Additional results for clustered FL are shown in Appendix A.

6.5.4 Performance measures of the proposed clustering pipeline

Lastly, we include some brief performance measures of the clustering mechanism that was proposed in this work. We acknowledge that, given the streaming context of our application, the total execution time cannot be measured since we expect the data to be unbounded. However, all the components of our pipeline rely on the use of tumbling windows of data. Because of this, we present performance metrics measured for various windows.

The following experiments on the proposed online algorithm have parallelism equal to 8, meaning the computation is distributed among 8 different threads. The results are obtained by averaging the execution time of running the algorithm on the fixed-sized windows for each step of the pipeline. The design has been implemented in Apache Flink. For comparison, we also include the execution time obtained by an offline library implementation in Python with one execution thread.

Table 6.14 shows the results of this study. The offline version is a single-threaded Python implementation. The measurements for our online versions are approximate because, at the time of conducting this study, the tumbling windows cannot be configured to use the event time in the Iterative Data Stream model of Apache Flink. Thus, the amount of data entering the window cannot be precisely controlled. For the pattern matching step, the observed decrease in execution time is to be expected in the online version, since the execution is mainly distributed among different workers. However, we notice that the execution time of the k-means online implementation is much higher than the offline version. In this case, we are paying the price of using a Bulk Synchronous Processing model. The workers have to be synchronized after each iteration. Moreover, the workers reach consensus by exchanging messages, which also introduce a communication overhead in the system.

Table 6.14: Performance measures of the proposed clustering mechanism.

	Offline		**Online (approx.)**	
	1 day (1890 samples)	**7 days (13230 samples)**	**1 day (1890 samples)**	**7 days (13230 samples)**
Streaming k-means	0.214 sec	1.398 sec	3.297 sec	21.475 sec
Pattern matching	0.366 sec	14.4 sec	0.115 sec	0.278 sec

It might be the case that our datasets are not big enough to capture the increase in performance brought by parallelizing the computation. We believe further experiments with larger datasets should be conducted as part of future work.

6.6 Related Work

We discuss similar approaches found in the literature and compare them to our proposed pipeline. To the best of our knowledge, there is no other work that aims to investigate the problem of user diet and health forecasting in a streaming context with privacy guarantees. Thus, in the following subsections, we present works that are comparable to each of the steps chosen in our pipeline.

6.6.1 Multivariate Time-Series Clustering Mechanism

Our work is inspired by [183] whose clustering technique is composed out of two logical steps, discretization to univariate time series and clustering the resultant data. We adapted their framework to implement a streaming variant of for the first step – discretization, followed by computing the Hamming similarity for the second step – pattern matching. Since pattern matching is trivial compared to the discretization process, we focus on the related work for the latter.

Discretization of time-series. Our discretization process is very similar to the one proposed by [184]. The goal is to map each multivariate data point of a time-series into a discrete value. According to the authors, the shape of the pattern is the most important trait of a series instead of actual values. In their study, the real-valued time-series was split into windows. Each subsequence is clustered using k-means. Discretized version of the time-series is obtained by using the ID of the cluster with the closest centroid to each subsequence. Lastly, the series of symbols are clustered using a suitable similarity measure.

The symbolization method proposed by [184] is suitable for our case for two main reasons: (1) working with low-dimensional symbols decreases the execution time of the algorithm, which is crucial in low-latency systems such as streaming applications, and (2) working with symbols instead of raw data brings benefit in terms of stronger user privacy.

Our work is also inspired by the work proposed by [185], where the authors aimed to discover similar patterns in the traveling habits of the subjects over streaming trajectories. Their approach partitions the stream into windows applies a clustering algorithm to observe the movements of the individuals in the given time frame and uses the clusters to detect co-movement patterns. The authors cluster the locations with respect to a specific threshold and we follow the same principle with a few tweaks that suit our problem.

6.6.2 Approaches for Federated Learning

We focus on studies that aim to solve two of the drawbacks of FL: the shortcomings of training only one federated model that should provide accurate predictions for all the participants, and the need for additional privacy methods so that sensitive data cannot be inferred from the updates sent to the server. We organize our discussion around these two problems.

Clustered federated learning. Clustered Federated Learning framework [186] was introduced to improve the performance of classic federated algorithms by leveraging natural groups that exist in the client population. The concept has been adopted by several studies, such as [187], who applied a clustering technique on patients' data to boost the performance in predicting the hospitalization time and mortality using electronic medical records. Moreover, [188] studied time-series forecasting and showed that training separate global models for different clusters of time-series improves the performance.

This procedure is not very different from the classic FL. The server clusters the clients according to a chosen similarity metric. Let us assume it discovers k groups. Then, the FL algorithm proceeds with the small modification that the server maintains k models instead of one. When an update from a client arrives, the server must first check the cluster label of the client to decide which model the user's update will contribute to. Also, each client has to download the model corresponding to its cluster. We adapted this approach for our work.

Differentially private FL. Federated learning made a significant step towards better privacy protection by offering a training mechanism that can learn from clients' data without having access to the actual data. However, sensitive information can still be inferred from the updates sent to the central server. Zhu et al. [189] proved in their study that it is, in fact, possible to obtain such information from shared gradients.

Other aggregation algorithms only use the weights obtained from training the local model. Studies show that this technique is not safe either and that private information of individuals can still be divulged [190, 191]. Regardless of the parameters chosen to be shared, each new communication round in an FL algorithm can lead to data leaks, which accumulate in time throughout the process. DP can, thus, be used to conceal the contribution of each client during the training process. Similar to the general idea of DP, a trade-off must be found between privacy loss and model performance.

Geyer et al. [182] address the problem of differentially private FL from a client point of view. Their algorithm distorts the updates sent at each communication round by adding Gaussian noise. This method can maintain the privacy of the clients with only a small decrease in performance, given the dataset size is large enough. Similarly, [192] tackle the same problem in the context of sensitive health data. In their work, the authors add noise to the optimization function and prove this approach offers good data privacy while maintaining an adequate model performance. We applied this DP technique and studied its effect on the prediction accuracy of the model.

6.6.3 Noisy Data with LSTMs for Forecasting

We also study the effect of another DP method, which involves disturbing the training datasets themselves by adding noise. The study that served as our main inspiration for this DP algorithm has been undertaken by [193], in which the authors used DP for stock price predictions. Albeit in a non-federated context, this study is closely related to our work because it noises the data directly to achieve better privacy preservation.

6.7 Summary

We provide an online system that can forecast the dietary habits and health data of users of fitness tracking applications and/or wearable devices. To this extent, we have designed and implemented a pipeline capable of accurately predicting user behaviour and that can leverage similarities between individuals to improve model performance while guaranteeing data privacy. Depending on the dataset and features, our predictions are no more than 0.025% far off the ground-truth value with respect to the range of the value. Moreover, our clustering mechanism leverages similarity between the users to improve prediction accuracy while reducing the model training time, with up to 49% error reduction as compared to an FL model trained for the whole dataset. With high privacy guarantees on user data $\varepsilon = 0.1$, we show that the baseline model has a small drop in prediction accuracy and that data noising mechanism benefits from user clustering. Our clustering system manages to sustain the prediction accuracy and, in most cases, improve it, with a reduction of 38% error in prediction accuracy as compared to the baseline noisy data model in the best case. Our work has the following limitations: our clustering mechanism is currently an offline learning mechanism as it only assigns the k and the user clusters once. Hence, the clustering mechanism can not accommodate to changes that might appear in the real-time scenario such as: major changes in the user meal patterns due to external factors such as seasonal or location changes, addition of a large number of users that possess a unique meal pattern not well-represented by the already present meal clusters, or addition of new features to the system; all these changes might require recomputing the k for streaming k-means clustering followed by the pattern matching mechanism to find new user clusters. Hence, for future work, we consider investigating adaptive k-means and online ML models. We believe adaptive modeling will help in improving the performance of the system.

Acknowledgment

This work is done in collaboration with Sonia-Florina Horchidan and Zainab Abbas at KTH. I would like to thank Zainab for her great ideas for implementing the time-series forecasting pipeline and for co-supervsion of our master student Sonia, and to Sonia for her hard work in implementing the pipeline for time-series forecasting. We are also thankful to Asst. Prof. Paris Carbone and Prof. Sonja Buchegger at KTH for their valuable advice, guidance and support during this research project.

CHAPTER

7

PyDPLib: Python Differential Privacy Library for Private Medical Data Analytics

7.1 Introduction

Pharmaceutical and medical technology companies are interested in accessing real-world medical data for driving their business models and decisions, which aids in the provision of continuously improving personalized services. However, these companies are not interested in personally identifiable data or protected health information (PHI) of the individuals but rather the cohort data. These companies cooperate with medical institutions who collect the medical data and want to share it, but they need to protect the privacy of the participating individuals.

Sharing data with PHI raises privacy concerns due to the threat of misuse or re-identification. Data protection laws like the EU's General Data Protection Regulation (GDPR) ensure higher public trust in data sharing, and enforce informed use of collected user data by the companies. GDPR enforces *privacy-by-design*, which means that the technology is designed with data privacy preservation [13]. One solution is to use information flow control [3] and design privacy-centric platforms that possess data flow models with respective permissions for every type of user to ensure user privacy and transparent accountability. However, a privacy-centric platform alone cannot solve the problem of sharing medical data with medical technology companies.

Data anonymization is commonly employed for privacy preservation in the health care industry, although anonymization alone does not guarantee sufficient privacy preservation due to the risk of re-identification and attribute disclosure [11]. A possible solution could be the use of realistic synthetic datasets for enhanced user privacy with reduced risk of re-identification. However, generation of mass-scale synthetic datasets might not always be reasonable due to: 1) the need for continuous integration and generation of new data, and 2) fixed privacy preservation levels for all parties. Mechanisms for secure multiparty computation [194] are also proposed in literature for collectively computing private statistics without sharing the sensitive data. However, these solutions might be computationally expensive and may not always cater to solving the problem of sharing statistical patterns with third parties.

Differential privacy (DP) [4] allows computing statistical patterns in a dataset while withholding information about individuals in the dataset. DP provides provable privacy guarantees and ensures the minimal impact of participation of a single individual in a database, by adding calculated noise depending on the queried statistical patterns. DP can also cater to flexible noise addition based on the user type and privilege. Numerous DP libraries exist in literature such as: Google's DP library [195] and its python wrapper *PyDP* [196], *Diffprivlib* IBM's DP library [197] and open-source implementation [198], *dp-stats* library for differentially private statistics and machine learning algorithms [199], mechanisms for DP histogram publication [200], and *OpenDP* collection of tools for statistical analysis of sensitive private data [201]. However, to the best of our knowledge, most of these tools are end-to-end solutions for computing DP statistical measurements, which makes it difficult to integrate them in an already-existing system without making significant changes to the system internals. Using these DP tools also requires an understanding of privacy preservation techniques which might be difficult for professionals in the medical domain. Moreover, these tools cannot often be used in conjunction with data visualization software, which makes it hard to integrate them in end-to-end systems that only require implementation of a privacy preservation layer.

We have developed PyDPLib, a platform-independent differential privacy library in Python. We have also developed a privacy-centric platform for structured data collection that employs PyDPLib, and we show our results on a database of prostate cancer patients. Our reporting software *SmartReports* and a PI-RADS [202] template was used to collect this dataset. Moreover, by using a software with interactive visualization such as plotly [203], PyDPLib is used to create interactive and private statistical plots.

The main contributions of this chapter are as follows.

- We have developed a differential privacy library PyDPLib for computing private statistics with medical data as a use case.
- PyDPLib provides different levels of privacy preservation with noise addition guarantees, depending on the user type and privilege.
- Our reporting software *SmartReports* uses information flow control, and when used with PyDPLib, ensures 2-layer privacy preservation.

- PyDPLib supports a variety of statistical operations on continuous and discrete data, and can be used in conjunction with any data visualization software.
- We have developed a template for structured clinical data collection and curated a human expert labeled dataset based on our template. We demonstrate our results for PyDPLib on this dataset.

7.2 Information Flow Control

Information flow control (IFC) tracks how information propagates through a program during execution to make sure that the program handles the information securely [3]. IFC is used as a privacy preservation technique by creating data flow models and specifying data flow policies. The creation of data flow models with respective data flow policies (permissions) ensures user privacy and transparent accountability. Privacy-centric platform is one of the solutions based on IFC. This solution requires information flows and privileges to be declared beforehand, so all the data elements are attached to respective data policies [67, 104, 105].

7.3 Structured Clinical Data Collection

SmartReports Reporting Software. SmartReports enables structured reporting of radiological examinations and procedures evaluated using flexible decision trees. Furthermore, the underlying data is machine-readable and therefore does not require reverse-engineering with techniques like natural language processing. The generated reports are exported to our electronic data capture (EDC) software, where the data is accessible for processing by PyDPLib.

Electronic Data Capture. Our EDC web application is a privacy-centric platform implementing *privacy-by-design* principles of the GDPR. State-of-the-art data protection measures have been adopted based on the Open Web Application Security Project (OWASP) Top Ten list, which represents a broad consensus about the most critical security risks. The clinical report data is imported and stored in this application and the DP analyses are generated server-side and can be accessed by users based on their data flow policy.

PI-RADS Template. Clinical report data is captured for prostate cancer patients who underwent MR-imaging using the Prostate Imaging Reporting & Data System (PI-RADS) v1 [202]. PI-RADS has several important objectives, among them the global standardization of acquisition, interpretation and reporting of prostate multiparametric MRI (mpMRI), simplification and standardization of report terminology and content, assistance in selection of patients undergoing biopsy and patient management, and facilitation of large scale clinical trials for data collection, outcome monitoring, and further research. It is designed to improve patient outcomes by detecting clinically significant prostate cancer and reduce the need for biopsies and unnecessary treatment. The PI-RADS v1 definition of clinically significant cancer (based on pathology/histology) is: Gleason score ≥ 7 (including $3 + 4$ with prominent, but not predominant Gleason 4 component) and/or volume $\geq 0.5ml$ and/or extra prostatic extension (EPE). Table 7.1 shows the 5-point scale for the PI-RADS score based on the likelihood of combination of mpMRI findings correlated with the presence of a clinically significant cancer. It is applied to each lesion. In addition to the PI-RADS score, information regarding age, Gleason score, prostatitis, benign prostatic hyperplasia (BPH), number of lesions and therapy is also collected.

Table 7.1: 5-point scale of PI-RADS assessment score

Score	**Assessment**
1	Very low (clinically significant cancer is highly unlikely)
2	Low (clinically significant cancer is unlikely)
3	Intermediate (clinically significant cancer is equivocal)
4	High (clinically significant cancer is likely)
5	Very high (clinically significant cancer is highly likely)

7.4 PyDPLib : Differential Privacy Library

PyDPLib provides ε-DP with Laplacian noise addition. The most common usages of DP mechanisms add DP-noise to the output of the computed statistical measures. However, since our platform provides a visual representation of both the raw and aggregated statistics, PyDPLib adds noise to the input data points which may or may not be aggregated. PyDPLib uses *Numpy* for dependencies, and supports a wide range of statistical queries and data types.

Figure 7.1 shows a high-level diagram displaying inputs/outputs and the interactions between different methods in PyDPLib. Input from the database or reporting system is passed to $dp\_calculate$ method. The $dp\_calculate$ first sets the noise level based on the input `noise_factor`. Afterwards, $dp\_calculate$ calls the $Binary\ Search$ method to find a suitable ε for the desired noise settings and S_f, by using the $Laplace\ mechanism$ for noise addition. This binary search is done in recursion until we find an ε that gives us the appropriate noise percentage. Finally, ε is used to noise the input data and PyDPLib returns the differentially private data dp_data, ε, S_f as well as the percentage noise.

7.4.1 Threat Model and Types of Users

In our approach, users are divided into three types:

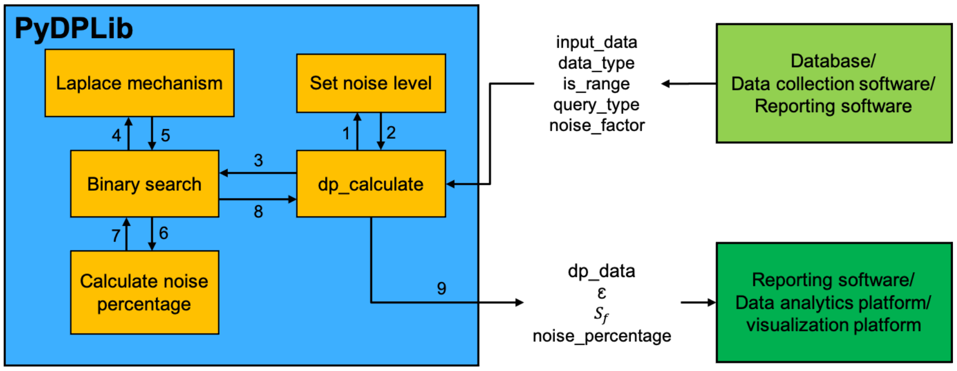


Figure 7.1: PyDPLib methods and interaction with other modules.

Owner: Hospital that own the data, and possess the full right to information disclosure;

Collaborator: Collaborating hospitals that access the data for research and other collaborative services;

Third party: includes commercial partners that need access to the medical data without information disclosure.

For our threat model, the data owners are trusted parties. They are also data curators and assign the data flow policies. The collaborators are trusted but can be *honest-but-curious* parties in the worst case. Therefore, we recommend using privacy preservation for collaborators. Third parties should not have access to the data but could visualize the underlying information to make general observations on the data patterns. In summary, third parties should not be able to extract the information of a single user based on the visualizations.

7.4.2 Setting the Appropriate Noise Factor

Depending on the type of user as mentioned earlier, PyDPLib selects respective noise addition settings. The Laplacian noise addition is controlled by the `noise_factor`, which offers three settings for noise addition to input data:

- 0: **low**. $0 - 5\%$ noise addition. Suitable for data owners (privileged users).
- 1: **medium**. $5 - 10\%$ noise addition. Suitable for collaborators requiring a low loss of accuracy.
- 2: **high**. $10 - 20\%$ noise addition. Suitable for third parties requiring statistics without data disclosure.

PyDPLib sets the appropriate noise level based on `noise_factor`. Afterwards, PyDPLib selects the appropriate ε for the Laplacian noise mechanism depending on the range, data type, and sensitivity of the statistical query S_f. In case of `is_range` or categorical data, the data is first fit to the input range and afterwards, noise percentage is calculated. We use a binary search algorithm for ε selection.

Algorithm 4 Binary Search for ε

```
Data: epsilons, $low_{idx}$, $hi_{idx}$, $noise\_level$, $input\_data$, $is\_bool$, $is\_range$, $S_f$
Result: $\varepsilon$
Initialize $sum$ to 0.  Set $noise\_lvl\_low$ and $noise\_lvl\_high$ depending on
  $noise\_level$.
if ($hi_{idx} \geq low_{idx}$) then
    $mid = (hi_{idx} + low_{idx})/2$
    for 5 times do
        for datum in $input\_data$ do
            Calculate noised data using Laplacian noise with $S_f$ and $\varepsilon =$
              $epsilons[mid]$
        end
        if $is\_range$ then fit noised data to range;
        Calculate percentage noise and add to $sum$.
    end
    Set $noise\_mid$ to the average of $sum$.
    if $noise\_lvl\_low \leq noise\_mid \leq noise\_lvl\_high$ then
        return $\varepsilon = epsilons[mid]$;                    /* found */
    else if $noise\_mid < noise\_lvl\_low$ then
        Repeat Binary Search in $[low_{idx}, mid]$
    else
        Repeat Binary Search in $[mid, hi_{idx}]$
else
    return 0 ;                                                /* not found */
end
```

7.4.3 Binary Search Algorithm for ε Selection

We perform a binary search for ε based on the `noise_factor` and the sensitivity S_f of the statistical query. low_{idx} and hi_{idx} are used to query low and high indexes in the epsilons array respectively. The binary search algorithm also takes additional parameters for determining data fitting mechanisms and noise calculation: `is_bool` and `is_range` are used for data fitting, and $noise\_level$ indicates the noise margins for selected noise factor. Algorithm 4 shows the complete Binary Search algorithm. Once an appropriate ε is found, the data points are noised according to S_f of the selected statistical query, and forwarded to the output along with selected value of ε.

7.4.4 Supported Data Types

PyDPlib supports a vast variety of data formats as well as continuous and discrete (or categorical) data. The categorical variables must be mapped to a numeric format. `data_type` could be char, string, int, numpy.int, float, numpy.float, bool, and numpy.bool. `is_range` (True/False) specifies if data values lie within a specific range. The output data is fitted to the range of input data after noise addition.

Fitting Data to Range. PyDPLib features the `fit_data_to_range` and `data_range` methods to handle categorical variables or range-bound data. The noising mechanism must return valid values regardless of the chosen noising levels. When `is_range` is `True`, the range of valid data points is inferred from the input data (non-noised) using the method `data_range`. This method returns all the unique data points as `range_values`. Afterwards, `fit_data_to_range` takes the following input parameters.

- `data`: noised data to fit to range;
- `range_values`: unique and valid data point values inferred from input data;
- `data_type`: could be int, numpy.int, float, numpy.float, bool, and numpy.bool.

The fitted result is interpreted as:

$$x'_{fitted} = \frac{(max_r - min_r)(x' - min_{data})}{max_{data} - min_{data}} + min_r \tag{7.1}$$

where max_r and min_r are the maximum and minimum input values inferred from `range_values`, and x' is a noised data point in `data` with maximum and minimum noised values given by max_{data} and min_{data} respectively.

Continuous data is represented by float or numpy.float, and the fitting mechanism is as in Equation 7.1. `fitted_data` is used to compute the noise percentage as compared to the input data. For boolean data, the noised data is fitted as: `True` if noised data > 0.5, else `False`. This is in agreement with the randomized response for DP in boolean attributes as observed in [204]. An alternative approach for noising the boolean values independently of the statistical query is to use the coin flip algorithm, where the reported boolean value is dependent on the outcome of the successive coin flips. However, repeated queries may expose the original probabilities of the underlying data.

7.4.5 Supported Statistical Queries and Sensitivity

PyDPLib offers four statistical `query_type`: 1) count or histogram, 2) average or mean, 3) median, and 4) variance. The noise addition mechanism (Equation 2.7) uses sensitivity S_f of the statistical operation to compute the margin of added noise, and each data point is individually noised. If the input data is categorical or range-bound, the noised data point is then fitted to the input range accordingly. Sensitivity of statistical queries for calibrating the noise addition is a well studied problem in literature [4,30,205,206]. We now describe the S_f of each statistical query in detail.

Count or Histogram. Count or histogram queries are the simplest of statistical operations as the addition or deletion of a single individual or record can change the count by at most 1 [4]. According to Equation 2.6, the sensitivity for count or histogram query is given as $S_{f(hist)} = 1$. Therefore, noise is sampled from $Lap(0, S_{f(hist)}/\varepsilon)$ distribution.

Average or Mean. For computing the sensitivity of average or mean, we need to compute the upper and lower bounds of the input data. For n input data points with lower bound a and upper bound b, the sensitivity is given by [207]:

$$S_{f(avg)} = |b - a|/n \tag{7.2}$$

Each data point is individually noised with a randomly picked sample from $Lap(0, S_{f(avg)}/\varepsilon)$ distribution, and a noisy average is computed by the data analytics software.

Median. Although the median and mean queries are statistically different, they exhibit the same sensitivity. The sensitivity of median query on an input data with n entries with lower bound a and upper bound b is given by [205]:

$$S_{f(med)} = |b - a|/n \tag{7.3}$$

x' is computed by adding a randomly picked sample from $Lap(0, S_{f(med)}/\varepsilon)$ distribution to x data point.

Variance. Sensitivity of variance for input data with n entries with lower bound a and upper bound b is given by:

$$S_{f(var)} = (b - a)^2/n \tag{7.4}$$

Here x' is computed by adding a randomly picked sample from $Lap(0, S_{f(var)}/\varepsilon)$ distribution to x data point. The noised data is then fitted to the input range, if applicable.

7.5 Experiments and Results

920 prostate cancer patient reports based on the PI-RADS template have been collected within the years 2017-18 by board-certified radiologists, and manually labelled by domain experts. Some of the collected and manually extracted attributes are shown in Table 7.2. We illustrate a variety of private statistical plots on the PI-RADS dataset using PyDPLib and visualized with Plotly [203]. Since PyDPLib perturbs the data distribution according to the data type, range and desired statistical query; and is independent of the used plotting mechanism, any data analytics or visualization software can be used to compute and display these plots.

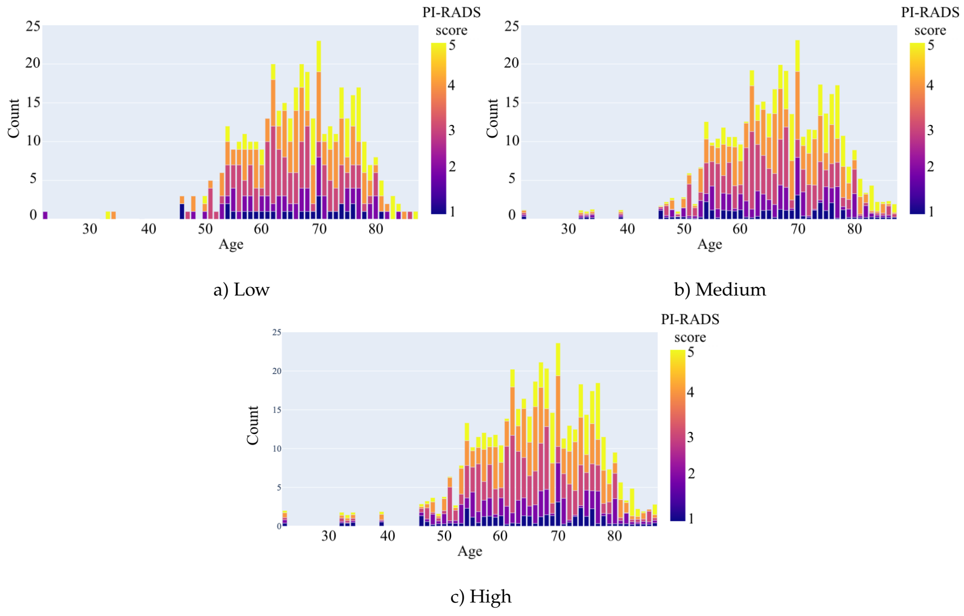


Figure 7.2: Histograms for patient age vs. PI-RADS scores with different `noise_factor` settings (low – high). Example for x-versus-y plot with noised y-axis, `query_type` = 1, `is_range = True`, and `data_type = int`.

`query_type` 1: Histograms. Figure 7.5 shows the histogram of the patient ages. Due to privacy concerns, we only illustrate the histogram in high noise settings. `is_range` is True as age can only be positive. Ages are directly noised using PyDPLib and displayed as a histogram. An alternative approach would be to compute the counts first and pass them to PyDPLib, as we discuss for the following histograms.

Chapter 7.2 shows the histogram plots for age versus PI-RADS score in low, medium and high privacy settings. Absolute ages are used, and `is_range = True`

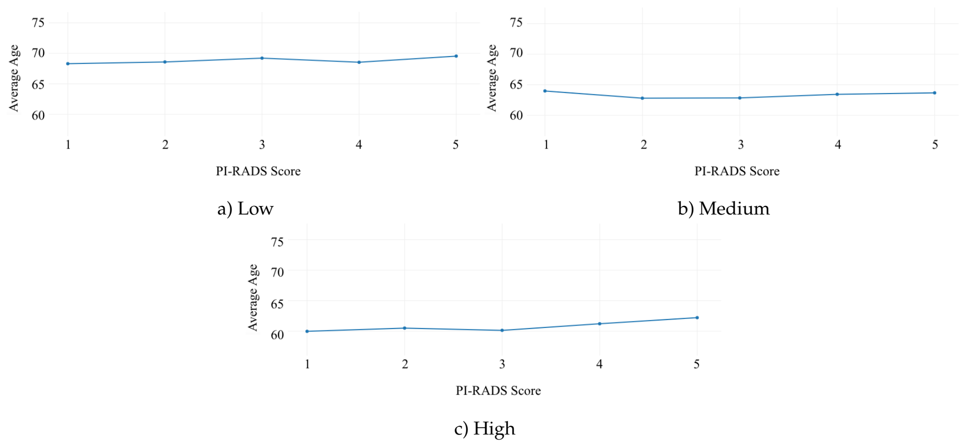


Figure 7.3: PI-RADS scores vs. average patient age with different `noise_factor` settings (low – high). Example for x-versus-y plot with noised x-axis, `query_type` = 2, `is_range = True`, and `data_type = float`.

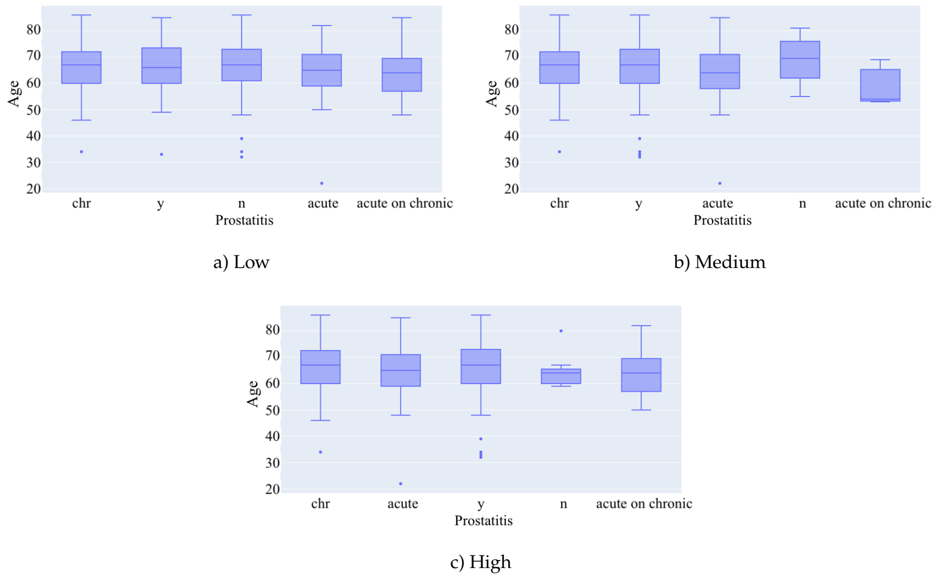


Figure 7.4: Prostatitis vs. median age with different `noise_factor` settings (low – high). Example for x-versus-y plot with noised x-axis, `query_type` = 3, `is_range = True`, and `data_type = int`.

Table 7.2: PI-RADS dataset

Attribute	Type	Data type
Anonym_ID	numerical	int
RequestRadiologyDepartmentCode	categorical	string
ProcedureValidatingPhysicianName	free text	string
LastModified	categorical	date
Age	numerical	float
Year	numerical	int
Gleason score	categorical	int
PCA_no_PIRADS_defined	categorical	bool
Follow up	categorical	string
PIRADS score	categorical	int
Number_of_Lesions	numerical	int/string
Prostatitis	categorical	string
BPH	categorical	bool
Primary_TU_LocalRecurrence_or_Progress	categorical	string
Therapy	categorical	string
TURP surgery	categorical	string

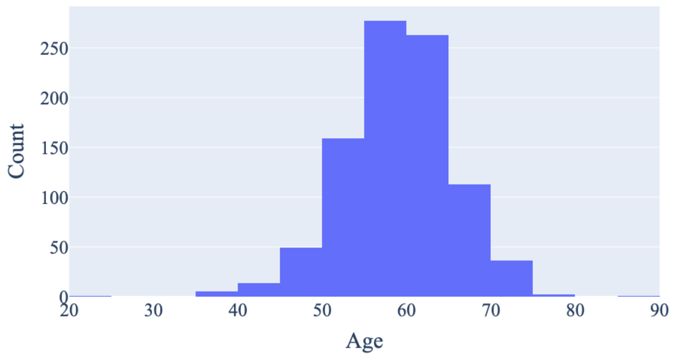


Figure 7.5: Age histogram with `noise_factor = 2` (high). Example for `query_type` = 1, `is_range = True`, and `data_type = float`.

as count can only be positive. Moreover, PI-RADS score can only have valid values in the range $1 - 5$ as described in Section 7.3. The counts for PI-RADS score for each age are computed on raw data, and passed to the PyDPLib. This is an example of x-versus-y plot, where only y-axis is noised. As can be seen, the overall distribution of samples is maintained in all noise settings. However, the individual frequencies recorded in the histograms are noised differently depending on the selected noise factor for increased user privacy. This makes it hard to single out an individual from the visual plot. Prostate cancer occurrence becomes high in ages $50 - 80$, with the highest levels of PI-RADS scores in $60 - 70$'s. This behaviour is generally observed regardless of the noise factor.

`query_type` 2: Average or mean. Chapter 7.3 shows the PI-RADS score versus the average patient age in low, medium and high privacy settings. Ages are interpreted as continuous data with float data type, and `is_range` is set to `True` since the ages can only be positive. Patient ages are individually noised by PyDPLib and noisy average is computed. This is an example of x-versus-y plot where only x-axis is noised. As can be seen, the average age for all PI-RADS scores lies in the $60 - 70$'s range. Addition of a new record with an extreme value will alter the query sensitivity, and PyDPLib will select an ε that offers the desired noise addition with low impact on overall distribution.

`query_type` 3: Median. Chapter 7.4 shows the median patient age versus Prostatitis. As shown in Table 7.2, Prostatitis is a categorical attribute (`is_range = True`) with values: `chr` (chronic), `y` (yes), `n` (no), `acute` and `acute on chronic`. These categorical variables are mapped to numerical values and passed to PyDPLib. After noising, these numerical values are mapped to original categories and visualized as a box plot with median bars. The resultant plot is an example of x-versus-y plot with noised x-axis. Alternatively, we can noise the ages and visualize them with respect to Prostatitis. Similarly, we can create many interesting statistical plots for various data types by using PyDPLib with any visualization software.

7.6 Summary

Medical institutions that want to share cohort statistics with external parties, e.g. pharmaceutical companies, have to protect the privacy of the individual patients that contributed to the underlying data. We have presented PyDPLib, a differential privacy library for private medical data analytics that greatly simplifies this task. PyDPLib offers different common statistical operations, such as histograms and mean, and noises the input data according to the type of operation, the data type, and the privilege level of the intended end user. The limitations of PyDPLib include: the current version requires more testing for handling the boolean data as the test database did not have many boolean features to experiment with. Moreover, all the categorical variables must be mapped to a numerical format for processing by PyDPLib. For future work, the library can be expanded to support more complex statistical queries and ML models in order to be used for displaying a wider range of the statistics on large-scale private ML models.

Based on a use case with real data, we have shown that PyDPLib allows creating statistical data visualizations that preserve the underlying data distributions without compromising the privacy of the individuals. Although developed for concrete use in our reporting platform, PyDPLib is general enough to be used in any data analytics or visualization application that requires differential privacy.

Acknowledgment

This work is done during the author's internship at Smart Reporting GmbH, Germany. We are thankful to Philipp Matthies and Francisco Pinto at Smart Reporting for their valuable suggestions, guidance and support during this research project. Their constant feedback helped us making this work successful. We are also truly grateful to our collaborators from Medical Faculty Mannheim, Heidelberg University, Mannheim, Germany, for providing the data for this study.

CHAPTER

8

Conclusion

In this thesis we have performed an empirical study on the impact of privacy preservation techniques on the different components of the IoT ecosystem with the smart health care domain as a use case. First, we presented a taxonomy and analysis of the privacy preservation techniques for the IoT ecosystem. Second, we presented a method to generate realistic synthetic and private smart health care datasets. Third, we presented, implemented and evaluated an end-to-end pipeline for machine learning with reconfigurable privacy on resource-limited computing devices which found a trade-off between privacy preservation, device resource usage and application accuracy. Afterwards, we presented and evaluated a solution for privacy-preserving forecasting of health care data streams with trade-offs between preserving user privacy, application runtime and prediction accuracy. Finally, we presented a solution for private data analytics in the form of a differential privacy library for private medical data analytics.

8.1 Summary of Results

The first part of this thesis presented in Chapter 3 provided an overview of privacy preservation techniques and solutions proposed so far in literature along with the IoT architecture layers (L1: perception, L2: network, L3: application) at which privacy is addressed by each solution, as well as their robustness to privacy breaching attacks. An analysis of functional and non-functional limitations of each solution was done, followed by a short survey of machine learning applications designed with these solutions. This taxonomy may serve as a guideline for selecting appropriate privacy preservation techniques according to the nature of data and the system performance requirements. We also identified open issues in the privacy preserving solutions when used in IoT environments. In general, there is no clear winner among the privacy preservation techniques. Depending on the system requirements, model obfuscation techniques, multi-tier and decentralized ML, private compute units and data flow models using blockchains and pre-defined information flows emerged as relatively strong candidates for privacy preservation. Another interesting observation was that industry and health care organizations often employ the relatively weaker solutions for privacy preservation due to their smaller negative impact on data utility and ease of use. Moreover, we noted that the solutions proposed in the recent years are trying to incorporate the GDPR, which will ensure better privacy guarantees for users.

Next, we designed, implemented and evaluate a solution for generating realistic synthetic private smart health care datasets from the sensitive non-private datasets to enable privacy preserving data sharing in Chapter 4. We proposed a generative adversarial network model coupled with differential privacy mechanisms for generating realistic and private smart health care datasets. Our solution catered to the unique challenges of smart health care data: *volume*, *velocity*, and *variety* - various data types and distributions. Our proposed solution not only enriched and augmented the input data samples but was also able to generate realistic synthetic data samples as well as generate the differentially private data samples under different settings: learning from a noisy distribution or noising the learned distribution. We tested and evaluated our proposed approach using a real-world Fitbit dataset. Our results indicated that our approach is able to generate high quality synthetic differentially private datasets that preserve the statistical properties of the original dataset.

In our next work presented in Chapter 5, we have designed, implemented and evaluated a solution for machine learning with reconfigurable privacy on resource-limited computing devices, complete with corresponding algorithms and a supporting end-to-end pipeline. The goal was to find a trade-off between preserving necessary user privacy, device resource usage and application accuracy. We used the generalization techniques for data anonymization and provided customized injective privacy encoder functions to make data features private. For this work, we assumed that regardless of the device resource availability, some data features must be essentially private. We termed all other data features that may pose a low privacy threat as the non-essential features. We proposed Dynamic Iterative Greedy Search (DIGS), a novel approach with corresponding algorithms to select the set of optimal non-essential data features to be private for machine learning applications provided device resource constraints. DIGS performed a breadth-wise iterative search for finding the most private version of data for the application provided device resource usage constraints, where all the essential features and a subset of non-essential features were made private on the edge device without resource overutilization. We evaluated DIGS on a Raspberry Pi model A device for an SVM-based classification application on real-life health care data. Our evaluation results showed that, while providing the required level of privacy, DIGS allows to achieve up to 26.21% memory, 16.67% CPU instructions, and 30.5% of network bandwidth savings as compared to making all the data private. Moreover, our chosen privacy encoding method demonstrated a positive impact on the accuracy of the classification model for our chosen application.

Chapter 6 presented the design, implementation and evaluation of an end-to-end pipeline for privacy preserving forecasting of data streams with a low loss of application accuracy. Our proposed pipeline used differential privacy and federated learning for the first time in the context of health care data streams and found a trade-off between preserving user privacy and achieving acceptable system performance in terms of application accuracy and runtime. We also proposed a clustering mechanism to leverage the similarities between users to improve the prediction accuracy as well as significantly reduce the model training time. Depending on the dataset and features, our predictions were no more than 0.025% far off the ground-truth value with respect to the range of value. Moreover, our clustering mechanism brought a significant reduction in the training time, with up to 49% reduction in prediction accuracy error in the best case, as compared to training a single model on the entire dataset. Moreover, our proposed privacy preserving mechanism at best introduced a decrease of $\approx 2\%$ in the prediction accuracy of the trained models. Furthermore, our proposed clustering mechanism reduced the prediction error even in highly noisy settings by as much as 38% as compared to using a single federated private model.

Finally, Chapter 7 presented the design and implementation of PyDPLib, a privacy preservation library for private data analytics and visualization with health care data as a use case. PyDPLib offers different common statistical operations, such as histograms and mean, and noises the input data according to the type of operation, the data type, and the privilege level of the intended end user. PyDPLib used a binary search algorithm to find the appropriate parameters for adding Laplacian noise to ensure differential privacy on the user data. We illustrated our results on a platform for visualizing private statistics using a database of prostate cancer patients. Our experimental results showed that PyDPLib allows creating statistical data plots without compromising patients' privacy while preserving underlying data distributions. Even though PyDPLib has been developed for use in a specific platform for reporting the radiological examinations and procedures, it is general enough to be used in any data analytics or visualization application that requires differential privacy.

We conclude from our aforementioned results that it is possible to find proper trade-offs between preserving necessary user privacy, retaining data utility and achieving acceptable system performance by means of applying (combinations of) appropriate privacy preservation techniques to machine learning based systems. Efficient and scalable solutions like differential privacy, federated learning and generalization can be used stand-alone or in combination with other privacy preservation solutions, in order to design solutions that offer privacy preservation guarantees with relatively low impact on system performance in terms of reduced model accuracy, slower runtimes, and increased device resource usage.

8.2 Generalization to Other Application Domains

Although our solutions were designed and tested particularly for the smart health care IoT domain, they are general enough to be extended to other application domains such as smart homes and smart cities. The proposed approach for generating realistic synthetic private datasets is powerful enough to learn a wide variety of data distributions, and minor changes to the generative adversarial network model architecture can generate data samples with simpler/more complex distributions. Moreover, our solution for machine learning with reconfigurable privacy preservation on resource-limited computing devices is domain agnostic, and can be used with more complex privacy preservation solutions as long as the solutions can be expressed as injective functions. Finally, our PyDPLib library for privacy-preserving data analytics can be used for other IoT application domains and with other visualization platforms to even create interactive differentially private plots.

8.3 Ethical and Social Aspects

Our research work puts the users' privacy first while designing all the solutions for the smart health care domain. We have taken proper measures to preserve the privacy of the volunteers who contributed their data for our study. All the participant users were informed beforehand of the data features we will be collecting, where they will be stored and how we will process them. Moreover, the participants were asked beforehand for any particular health conditions that might display unique behaviours in their diet or activity patterns, and therefore, participants with unique health conditions were not included in this study. Moreover, we tried to counter the data bias by having as much of gender balance as possible within the participant pools for each location. However, the exact gender split for participant pool is not disclosed for privacy preservation reasons. Similarly, the participant pool was carefully chosen to have an overall population age within a close range (around 20's-30's). None of the user data has been stored on cloud platforms or storage services, instead, the de-identified data has only been stored on the personal machine of the writer of this dissertation. We have taken careful measures to remove any identifiable information from the dataset before sharing with master thesis students, and have used peer-to-peer data sharing mechanisms that do not store any data on an intermediate machine during data transfer.

From the social perspective, our solution for smart health care data generation will enable wide-scale data sharing for open use in research, and encourage opportunities for research collaborations between the academia and health care industry. We hope that this work will also contribute to the improvement of physical health of common users by increasing their participation and trust in the usage of personal devices and applications for smart health care. During the course of this study we have also contributed to the improvement of physical health of participating users, who became more conscious of their dietary and activity patterns during this study. Most of the volunteers ended up buying smart health care trackers after this study and resolved to adopt a healthier lifestyle. In summary, all these solutions have made an effort towards improving user trust in the adoption of smart health care platforms by finding proper trade-offs between preserving user privacy and providing good quality of service.

8.4 Future Work

For future work, we would like to test our proposed solutions on bigger large-scale real-world smart health care datasets to observe the benefits of employing our solutions in terms of achieved performance and privacy preservation on a large scale. Our solution for smart health care data generation could be extended to other domains such as smart homes for generating private realistic synthetic smart home datasets that may otherwise contain highly sensitive user data in their original form. Moreover, our solution for privacy preserving time-series forecasting of user health data streams proposes use of a clustering mechanism that requires partial data transfer to a central system which may slightly impact the privacy preservation experienced at the user end. We can look into alternative solutions for clustering the users with minimal data transfer, and find mechanisms to conceal user cluster membership at the server end. Moreover, PyDPLib could be extended to support more statistical operations and data types, and even accommodate machine learning models with privacy preservation guarantees.

In summary, smart health care is a fast-growing IoT domain with its unique challenges such as the requirements for efficient and scalable privacy preservation techniques that offer high data utility and low loss of application accuracy. There is a huge room for improvement in solutions for privacy preservation in the smart health care domain, and many lessons to be learned from the current research in the traditional health care industry.

Bibliography

[1] K. Ashton, “That ‘internet of things’ thing,” *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[2] O. Mazhelis, E. Luoma, and H. Warma, “Defining an internet-of-things ecosystem,” in *Internet of Things, Smart Spaces, and Next Generation Networking*, S. Andreev, S. Balandin, and Y. Koucheryavy, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–14.

[3] D. Hedin and A. Sabelfeld, “A perspective on information-flow control.” *Software safety and security*, vol. 33, pp. 319–347, 2012.

[4] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[5] S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci, “Enhancing privacy and preserving accuracy of a distributed collaborative filtering,” in *Proceedings of the ACM conference on Recommender systems*. ACM, 2007, pp. 9–16.

[6] A. Padron and G. Vargas. Multiparty homomorphic encryption. Online: https://courses.csail.mit.edu/6.857/2016/files/17.pdf. Accessed 28 October 2021.

[7] L. Sweeney, “k-anonymity: A model for protecting privacy,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, pp. 557–570, 2002.

[8] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 3, 2007.

[9] N. Li, T. Li, and S. Venkatasubramanian, “t-closeness: Privacy beyond k-anonymity and l-diversity,” in *23rd International Conference on Data Engineering*. IEEE, 2007, pp. 106–115.

[10] V. Costan and S. Devadas, “Intel SGX explained.” *IACR Cryptology ePrint Archive*, vol. 2016, no. 86, 2016.

[11] S. Imtiaz, R. Sadre, and V. Vlassov, “On the case of privacy in the IoT ecosystem: A survey,” in *International Conference on Internet of Things (iThings)*. IEEE, 2019, pp. 1015–1024.

[12] A. Pekar, J. Mocnej, W. K. G. Seah, and I. Zolotova, “Application domain-based overview of iot network traffic characteristics,” *ACM Computing Surveys*, vol. 53, no. 4, Jul. 2020. [Online]. Available: https://doi.org/10.1145/3399669

[13] Privacy by Design | General Data Protection Regulation (GDPR). Online: https://gdpr-info.eu/issues/privacy-by-design/. Accessed 28 October 2021.

[14] B. Sudharsan, J. G. Breslin, and M. I. Ali, “Rce-nn: A five-stage pipeline to execute neural networks (cnns) on resource-constrained iot edge devices,” in *Proceedings of the 10th International Conference on the Internet of Things*, ser. IoT ’20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3410992.3411005

[15] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, “Apache Flink: Stream and batch processing in a single engine,” *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 36, no. 4, 2015.

[16] A. S. Foundation, “Apache storm,” Online: https://storm.apache.org/, 2021, accessed 28 October 2021.

[17] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, “Discretized streams: Fault-tolerant streaming computation at scale,” in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, ser. SOSP ’13. New York, NY, USA: Association for Computing Machinery, 2013, p. 423–438. [Online]. Available: https://doi.org/10.1145/2517349.2522737

[18] J. Wieringa, P. Kannan, X. Ma, T. Reutterer, H. Risselada, and B. Skiera, “Data analytics in a privacy-concerned world,” *Journal of Business Research*, vol. 122, pp. 915–925, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0148296319303078

[19] S. Imtiaz, M. Arsalan, V. Vlassov, and R. Sadre, “Synthetic and private smart health care data generation using GANs,” in *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021, pp. 1–7.

[20] S. Imtiaz, P. Matthies, F. Pinto, M. Maros, H. Wenz, R. Sadre, and V. Vlassov, “PyDPLib: Python differential privacy library for private medical data analytics,” in *2021 IEEE International Conference on Digital Health (ICDH)*, 2021, pp. 191–196.

[21] S. Imtiaz, S.-F. Horchidan, Z. Abbas, M. Arsalan, H. N. Chaudhry, and V. Vlassov, “Privacy preserving time-series forecasting of user health data streams,” in *IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 3428–3437.

[22] S. Imtiaz, Z. Tania, H. N. Chaudhry, M. Arsalan, R. Sadre, and V. Vlassov, "Machine learning with reconfigurable privacy on resource-limited computing devices," October 2021, to appear in the proceedings of 14th IEEE International Conference on Security, Privacy, and Anonymity in Computation, Communication, and Storage (IEEE SpaCCS 2021).

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[24] Fitbit. Online: https://www.fitbit.com/se/home. Accessed 28 October 2021.

[25] MyFitnessPal. Online: https://www.myfitnesspal.com/. Accessed 28 October 2021.

[26] I. Weber and P. Achananuparp. Myfitnesspal food diary dataset. Online: https://doi.org/10.13140/RG.2.2.14511.64167. Accessed 28 October 2021.

[27] S. Fedeli, F. Schain, S. Imtiaz, Z. Abbas, and V. Vlassov, "Privacy preserving survival prediction," December 2021, to appear in the proceedings of IEEE International Conference on Big Data (Big Data).

[28] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[29] R. D. Hjelm, A. P. Jacob, T. Che, A. Trischler, K. Cho, and Y. Bengio, "Boundary-seeking generative adversarial networks," *arXiv preprint arXiv:1702.08431*, 2017.

[30] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*. Springer, 2006, pp. 265–284.

[31] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *EUROCRYPT*. Springer, 2006, pp. 486–503.

[32] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Proceedings of STOC'09*, 2009, pp. 371–380.

[33] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference*. Springer, 2016, pp. 635–658.

[34] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *FOCS'10*. IEEE, 2010, pp. 51–60.

[35] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," in *ICML*, 2015, pp. 1376–1385.

[36] Q. Geng and P. Viswanath, "The optimal noise-adding mechanism in differential privacy," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 925–951, 2015.

[37] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 571–588, 2002.

[38] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[39] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*. PMLR, 2017, pp. 1273–1282.

[40] Z. Abbas, J. R. Ivarsson, A. Al-Shishtawy, and V. Vlassov, "Scaling deep learning models for large spatial time-series forecasting," in *IEEE Big Data*, 2019, pp. 1587–1594.

[41] T. W. Liao, "Clustering of time series data - a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[42] I. S. Dhillon and D. S. Modha, "A data-clustering algorithm on distributed memory multiprocessors," in *Large-scale parallel data mining, Workshop at SIGKDD*. Springer, 2002, pp. 245–260.

[43] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, pp. 157–66, 02 1994.

[44] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[45] M. Arsalan and A. Santra, "Character recognition in air-writing based on network of radars for human-machine interface," *IEEE Sensors Journal*, vol. 19, no. 19, pp. 8855–8864, 2019.

[46] Cisco. Global 2021 Forecast Highlights. Online: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf. Accessed 28 October 2021.

[47] K. Zhao and L. Ge, "A survey on the internet of things security," in *9th International Conference on Computational Intelligence and Security*, Dec 2013, pp. 663–667.

[48] Y.-K. Chen, "Challenges and opportunities of internet of things," in *17th Asia and South Pacific design automation conference*. IEEE, 2012, pp. 383–388.

[49] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," *arXiv preprint arXiv:1610.05755*, 2016.

[50] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The secret sharer: Measuring unintended neural network memorization & extracting secrets," *arXiv preprint arXiv:1802.08232*, 2018.

[51] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[52] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Symposium on Security and Privacy (IEEE S&P)*. IEEE, 2018, pp. 36–52.

[53] M. Juuti, S. Szyller, A. Dmitrenko, S. Marchal, and N. Asokan, "Prada: protecting against DNN model stealing attacks," *arXiv preprint arXiv:1805.02628*, 2018.

[54] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the CCS*. ACM, 2015, pp. 1322–1333.

[55] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[56] P. P. Jayaraman, X. Yang, A. Yavari, D. Georgakopoulos, and X. Yi, "Privacy preserving internet of things: From privacy techniques to a blueprint architecture and efficient implementation," *Future Generation Computer Systems*, vol. 76, pp. 540–549, 2017.

[57] A. Westin, *Privacy and freedom*. Atheneum New York, 1967, vol. 1.

[58] H. J. Smith, S. J. Milberg, and S. J. Burke, "Information privacy: measuring individuals' concerns about organizational practices," *MIS quarterly*, pp. 167–196, 1996.

[59] J. H. Ziegeldorf, O. G. Morchon, and K. Wehrle, "Privacy in the internet of things: threats and challenges," *Security and Communication Networks*, vol. 7, no. 12, pp. 2728–2742, 2014.

[60] J. Voelcker, "Stalked by satellite-an alarming rise in gps-enabled harassment," *IEEE Spectrum*, vol. 43, no. 7, pp. 15–16, 2006.

[61] N. Madaan, M. A. Ahad, and S. M. Sastry, "Data integration in IoT ecosystem: Information linkage as a privacy threat," *Computer law & security review*, vol. 34, no. 1, pp. 125–133, 2018.

[62] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 506–519.

[63] G. Kellaris, G. Kollios, K. Nissim, and A. O'neill, "Generic attacks on secure outsourced databases," in *Proceedings of the SIGSAC Conference*. ACM, 2016, pp. 1329–1340.

[64] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "LOGAN: Membership inference attacks against generative models," *Proceedings of Privacy Enhancing Technologies (PoPETs/PETS)*, pp. 133–152, 2019.

[65] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE S&P*. IEEE, 2017, pp. 3–18.

[66] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proceedings of the 22nd ACM SIGSAC Conference*. ACM, 2015, pp. 644–655.

[67] G. Sagirlar, B. Carminati, and E. Ferrari, "Decentralizing privacy enforcement for internet of things smart objects," *Computer Networks*, vol. 143, pp. 112–125, 2018.

[68] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the smart home: Privacy attacks and defenses on encrypted IoT traffic," *arXiv preprint arXiv:1708.05044*, 2017.

[69] A. Narayanan, J. Huey, and E. W. Felten, "A precautionary approach to big data privacy," in *Data protection on the move*. Springer, 2016, pp. 357–385.

[70] P. Ohm, "Broken promises of privacy: Responding to the surprising failure of anonymization," *UCLA l. Rev.*, vol. 57, p. 1701, 2009.

[71] J. Abowd, L. Alvisi, C. Dwork, S. Kannan, A. Machanavajjhala, and J. Reiter, “Privacy-preserving data analysis for the federal statistical agencies,” *arXiv preprint:1701.00752*, 2017.

[72] S. Milli, L. Schmidt, A. D. Dragan, and M. Hardt, “Model reconstruction from model explanations,” *arXiv preprint arXiv:1807.05185*, 2018.

[73] A. Fernandes, D. Cloete, M. Broadbent, R. Hayes, C.-K. Chang, R. Jackson, A. Roberts, J. Tsang, M. Soncul, J. Liebscher, R. Stewart, and F. Callard, “Development and evaluation of a de-identification procedure for a case register sourced from mental health electronic records,” *BMC medical informatics and decision making*, vol. 13, p. 71, 07 2013.

[74] C. Kushida, D. Nichols, R. Jadrnicek, R. Miller, J. Walsh, and K. Griffin, “Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies,” *Medical care*, vol. 50 Suppl, pp. S82–101, 07 2012.

[75] K. Moselle, S. Robertson, and A. Koval, “‘Real-World" de-identification of high-dimensional transactional health datasets.” *Studies in health technology and informatics*, vol. 257, pp. 319–324, 2019.

[76] P. Zhao, J. Li, F. Zeng, F. Xiao, C. Wang, and H. Jiang, “ILLIA: Enabling k-Anonymity-Based Privacy Preserving Against Location Injection Attacks in Continuous LBS Queries,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1033–1042, 2018.

[77] A. Gkoulalas-Divanis, G. Loukides, and J. Sun, “Publishing data from electronic health records while preserving privacy: A survey of algorithms,” *Journal of biomedical informatics*, vol. 50, pp. 4–19, 2014.

[78] P. Zhao, H. Jiang, C. Wang, H. Huang, G. Liu, and Y. Yang, “On the performance of k-anonymity against inference attacks with background information,” *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 808–819, 2019.

[79] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang, “(α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing,” in *Proceedings of the 12th ACM SIGKDD*. ACM, 2006, pp. 754–759.

[80] R. Wang, Y. Zhu, T.-S. Chen, and C.-C. Chang, “Privacy-preserving algorithms for multiple sensitive attributes satisfying t-closeness,” *Journal of Computer Science and Technology*, vol. 33, no. 6, pp. 1231–1242, 2018.

[81] C. Yin, S. Zhang, J. Xi, and J. Wang, “An improved anonymity model for big data security based on clustering algorithm,” *Concurrency and Computation: Practice and Experience*, vol. 29, no. 7, p. e3902, 2017.

[82] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[83] C. Dwork, A. Smith, T. Steinke, and J. Ullman, "Exposed! a survey of attacks on private data," *Annual Review of Statistics and Its Application*, vol. 4, pp. 61–84, 2017.

[84] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "On the connection between differential privacy and adversarial robustness in machine learning," *stat*, vol. 1050, p. 9, 2018.

[85] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1175–1191.

[86] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[87] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[88] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, 2019.

[89] M. Ammad-Ud-Din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *arXiv preprint arXiv:1901.09888*, 2019.

[90] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *arXiv preprint arXiv:1802.07876*, 2018.

[91] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks," *arXiv preprint arXiv:1812.00910*, 2018.

[92] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data." in *NDSS*, vol. 4324, 2015, p. 4325.

[93] E. L. Cominetti and M. A. Simplicio, "Fast additive partially homomorphic encryption from the approximate common divisor problem," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2988–2998, 2020.

[94] H. Zhou and G. Wornell, “Efficient homomorphic encryption on integer vectors and its applications,” in *Information Theory and Applications Workshop (ITA)*. IEEE, 2014, pp. 1–9.

[95] S. Bogos, J. Gaspoz, and S. Vaudenay, “Cryptanalysis of a homomorphic encryption scheme,” *Cryptography and Communications*, vol. 10, no. 1, pp. 27–39, 2018.

[96] A. C.-C. Yao, “Protocols for secure computations,” in *FOCS*, vol. 82, 1982, pp. 160–164.

[97] B. Mirzasoleiman, M. Zadimoghaddam, and A. Karbasi, “Fast distributed submodular cover: Public-private data summarization,” in *NIPS*, 2016, pp. 3594–3602.

[98] M. Mitrovic, M. Bun, A. Krause, and A. Karbasi, “Differentially private submodular maximization: Data summarization in disguise,” in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 2478–2487. [Online]. Available: https://proceedings.mlr.press/v70/mitrovic17a.html

[99] G. Ayoade, V. Karande, L. Khan, and K. Hamlen, “Decentralized IoT data management using blockchain and trusted execution environment,” in *International Conference on Information Reuse and Integration (IRI)*. IEEE, 2018, pp. 15–22.

[100] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, “Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability,” in *Proceedings of the 17th IEEE/ACM CCGrid*. IEEE Press, 2017, pp. 468–477.

[101] G. Zyskind, O. Nathan, and A. S. Pentland, “Decentralizing privacy: Using blockchain to protect personal data,” in *2015 IEEE Security and Privacy Workshops*. IEEE, 2015, pp. 180–184.

[102] X. Liang, J. Zhao, S. Shetty, and D. Li, “Towards data assurance and resilience in IoT using blockchain,” in *MILCOM*. IEEE, 2017, pp. 261–266.

[103] T. McGhin, K.-K. R. Choo, C. Z. Liu, and D. He, “Blockchain in healthcare applications: Research challenges and opportunities,” *Journal of Network and Computer Applications*, 2019.

[104] I. Zavalyshyn, N. O. Duarte, and N. Santos, “Homepad: A privacy-aware smart hub for home environments,” in *IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 58–73.

[105] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "Flowfence: Practical data protection for emerging IoT application frameworks," in *25th USENIX Security Symposium*, 2016, pp. 531–548.

[106] J. Yang, K. Yessenov, and A. Solar-Lezama, "A language for automatically enforcing privacy policies," in *ACM SIGPLAN Notices*, vol. 47, no. 1. ACM, 2012, pp. 85–96.

[107] Z. B. Celik, E. Fernandes, E. Pauley, G. Tan, and P. McDaniel, "Program Analysis of Commodity IoT Applications for Security and Privacy: Challenges and Opportunities," *arXiv preprint arXiv:1809.06962*, 2018.

[108] I. Ng, R. Maull, G. Parry, J. Crowcroft, K. Scharf, T. Rodden, and C. Speed, "Making value creating context visible for new economic and business models: Home Hub-of-all-Things (HAT) as platform for multisided market powered by IoT," in *Panel Session at The Future of Value Creation in Complex Service Systems Minitrack of Hawaii International Conference on Systems Science (HICSS)*, 2013, pp. 7–10.

[109] Hub-of-All-Things. Online: https://www.hubofallthings.com/. Accessed 28 October 2021.

[110] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on intel sgx," in *Proceedings of the 10th European Workshop on Systems Security*. ACM, 2017, p. 2.

[111] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "SoK: Security and privacy in machine learning," in *IEEE EuroS&P*. IEEE, 2018, pp. 399–414.

[112] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential privacy and machine learning: a survey and review," *arXiv preprint arXiv:1412.7584*, 2014.

[113] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[114] C. Dwork and V. Feldman, "Privacy-preserving prediction," *arXiv preprint arXiv:1803.10266*, 2018.

[115] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," *arXiv preprint arXiv:1803.05961*, 2018.

[116] S. Nijssen and E. Fromont, "Optimal constraint-based decision tree induction from itemset lattices," *Data Mining and Knowledge Discovery*, vol. 21, no. 1, pp. 9–51, 2010.

[117] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *IEEE S&P*. IEEE, 2017, pp. 19–38.

[118] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, 2009.

[119] F. Zhang, V. E. Lee, R. Jin, S. Garg, K.-K. R. Choo, M. Maasberg, L. Dong, and C. Cheng, "Privacy-aware smart city: A case study in collaborative filtering recommender systems," *Journal of Parallel and Distributed Computing*, 2018.

[120] R. Guerraoui, A.-M. Kermarrec, R. Patra, M. Valiyev, and J. Wang, "I know nothing about you but here is what you might like," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2017, pp. 439–450.

[121] S. Spiekermann and L. F. Cranor, "Engineering privacy," *IEEE Transactions on Software Engineering*, vol. 35, no. 1, pp. 67–82, Jan 2009.

[122] P. Voigt and A. Von dem Bussche, "The EU General Data Protection Regulation (GDPR)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[123] M. Veale, R. Binns, and L. Edwards, "Algorithms that remember: model inversion attacks and data protection law," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 376, no. 2133, p. 20180083, 2018.

[124] M. Young, L. Rodriguez, E. Keller, F. Sun, B. Sa, J. Whittington, and B. Howe, "Beyond open vs. closed: Balancing individual privacy and public accountability in data sharing," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM, 2019, pp. 191–200.

[125] N. Ahmad, R. P. George, and R. Jahan, "Emerging trends in iot for categorized health care," in *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, vol. 1, 2019, pp. 1438–1441.

[126] R. Lee, R. Jang, M. Park, G. Jeon, J. Kim, and S. Lee, "Making IoT data ready for smart city applications," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2020, pp. 605–608.

[127] R. Dagar, S. Som, and S. K. Khatri, "Smart farming – IoT in agriculture," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2018, pp. 1052–1056.

[128] N. Scarpato, A. Pieroni, L. Di Nunzio, and F. Fallucchi, "E-health-IoT universe: A review," *management*, vol. 21, no. 44, p. 46, 2017.

[129] J. Walonoski, M. Kramer, J. Nichols, A. Quina, C. Moesel, D. Hall, C. Duffett, K. Dube, T. Gallagher, and S. McLachlan, "Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record," *Journal of the American Medical Informatics Association*, vol. 25, no. 3, pp. 230–238, 2018.

[130] H. Li, L. Xiong, L. Zhang, and X. Jiang, "DPSynthesizer: Differentially private data synthesizer for privacy preserving data sharing," *Proc. VLDB Endow.*, vol. 7, no. 13, p. 1677–1680, Aug. 2014. [Online]. Available: https://doi.org/10.14778/2733004.2733059

[131] M. K. Baowaly, C.-C. Lin, C.-L. Liu, and K.-T. Chen, "Synthesizing electronic health records using improved generative adversarial networks," *Journal of the American Medical Informatics Association*, vol. 26, no. 3, pp. 228–241, 2019.

[132] H. Ping, J. Stoyanovich, and B. Howe, "Datasynthesizer: Privacy-preserving synthetic datasets," in *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, 2017, pp. 1–5.

[133] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Advances in Neural Information Processing Systems*, 2019, pp. 7335–7345.

[134] A. Torfi and E. A. Fox, "COR-GAN: Correlation-capturing convolutional neural networks for generating synthetic healthcare records," *arXiv preprint arXiv:2001.09346*, 2020.

[135] C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama, "GAN-based synthetic brain MR image generation," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE, 2018, pp. 734–738.

[136] J. Jordon, J. Yoon, and M. Van Der Schaar, "PATE-GAN: Generating synthetic data with differential privacy guarantees," in *International Conference on Learning Representations*, 2018.

[137] R. Torkzadehmahani, P. Kairouz, and B. Paten, "DP-CGAN: Differentially private synthetic data and label generation," in *Proceedings of the IEEE CVPR Workshops*, 2019, pp. 0–0.

[138] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," *arXiv preprint arXiv:1802.06739*, 2018.

[139] C. Xu, J. Ren, D. Zhang, Y. Zhang, Z. Qin, and K. Ren, "GANobfuscator: Mitigating information leakage under GAN via differential privacy," *IEEE*

Transactions on Information Forensics and Security, vol. 14, no. 9, pp. 2358–2371, 2019.

[140] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," *arXiv preprint arXiv:1706.02633*, 2017.

[141] Y. Qu, S. Yu, J. Zhang, H. T. T. Binh, L. Gao, and W. Zhou, "GAN-DP: Generative adversarial net driven differentially privacy-preserving big data publishing," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[142] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," in *Machine learning for healthcare conference*. PMLR, 2017, pp. 286–305.

[143] Nutritionix API. Online: https://developer.nutritionix.com/. Accessed 28 October 2021.

[144] F. J. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[145] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2017.

[146] M. Wedel and P. Kannan, "Marketing analytics for data-rich environments," *Journal of Marketing*, vol. 80, no. 6, pp. 97–121, 2016.

[147] H. Dev, T. Sen, M. Basak, and M. E. Ali, "An approach to protect the privacy of cloud data from data mining based attacks," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. IEEE, 2012, pp. 1106–1115.

[148] V. N. Inukollu, S. Arsi, and S. R. Ravuri, "Security issues associated with big data in cloud computing," *International Journal of Network Security & Its Applications*, vol. 6, no. 3, p. 45, 2014.

[149] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "Threats to networking cloud and edge datacenters in the internet of things," *IEEE Cloud Computing*, vol. 3, no. 3, pp. 64–71, 2016.

[150] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," 2015.

[151] F. Kulsoom, A. Vizziello, H. N. Chaudhry, and P. Savazzi, "Joint sparse channel recovery with quantized feedback for multi-user massive mimo systems," *IEEE Access*, vol. 8, pp. 11 046–11 060, 2020.

[152] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux, "Preserving privacy in collaborative filtering through distributed aggregation of offline profiles," in *Proceedings of the third ACM conference on Recommender systems*, 2009, pp. 157–164.

[153] I. Mazeh and E. Shmueli, "A personal data store approach for recommender systems: enhancing privacy without sacrificing accuracy," *Expert Systems with Applications*, vol. 139, p. 112858, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419305688

[154] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[155] guppy3.PyPI. Online: https://pypi.org/project/guppy3/. Accessed 28 October 2021.

[156] How many calories should you eat per day to lose weight? Online: https://www.healthline.com/nutrition/how-many-calories-per-day#average-calorie-needs. Accessed 28 October 2021.

[157] How many steps do I need a day? Online: https://www.healthline.com/health/how-many-steps-a-day. Accessed 28 October 2021.

[158] About Adult BMI:Healthy Weight, Nutrition, and Physical Activity | CDC. Online: https://www.cdc.gov/healthyweight/assessing/bmi/adult_bmi/index.html. Accessed 28 October 2021.

[159] How many calories are in one gram of fat, carbohydrate, or protein? | Food and Nutrition Information Center. Online: https://www.nal.usda.gov/fnic/how-many-calories-are-one-gram-fat-carbohydrate-or-protein. Accessed 28 October 2021.

[160] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *CoRR*, vol. abs/1605.09782, 2016. [Online]. Available: http://arxiv.org/abs/1605.09782

[161] J. Vitak, Y. Liao, P. Kumar, M. Zimmer, and K. Kritikos, "Privacy attitudes and data valuation among fitness tracker users," in *International Conference on Information*. Springer, 01 2018, pp. 229–239.

[162] E. Bertino, "Data privacy for IoT systems: Concepts, approaches, and research directions," in *2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 3645–3647.

[163] P. Zhao, H. Jiang, C. Wang, H. Huang, G. Liu, and Y. Yang, "On the performance of k-anonymity against inference attacks with background information," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 808–819, 2019.

[164] S.-F. Horchidan, "Real-time forecasting of dietary habits and user health using federated learning with privacy guarantees," Master's thesis, School of Electrical Engineering and Computer Science (EECS), KTH Royal Institute of Technology, Sweden, 2020.

[165] J. Orlosky, O. Ezenwoye, H. Yates, and G. Besenyi, "A look at the security and privacy of fitbit as a health activity tracker," in *Proceedings of the 2019 ACM Southeast Conference*, 04 2019, pp. 241–244.

[166] R. Dong, L. J. Ratliff, A. A. Cárdenas, H. Ohlsson, and S. S. Sastry, "Quantifying the utility–privacy tradeoff in the internet of things," *ACM Transactions on Cyber-Physical Systems*, vol. 2, no. 2, May 2018. [Online]. Available: https://doi.org/10.1145/3185511

[167] L. M. Aiello and G. Ruffo, "LotusNet: Tunable privacy for distributed online social network services," *Computer Communications*, vol. 35, no. 1, pp. 75–88, 2012.

[168] G. Skinner, "Dynamic user reconfigurable privacy and trust settings for collaborative industrial environments," in *INDIN'05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005.* IEEE, 2005, pp. 761–766.

[169] G. Ghinita, K. Nguyen, M. Maruseac, and C. Shahabi, "A secure location-based alert system with tunable privacy-performance trade-off," *GeoInformatica*, vol. 24, no. 4, pp. 951–985, 2020.

[170] C. Niu, F. Wu, S. Tang, L. Hua, R. Jia, C. Lv, Z. Wu, and G. Chen, "Billion-scale federated learning on mobile clients: a submodel design with tunable privacy," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[171] R. Narendula, T. G. Papaioannou, Z. Miklós, and K. Aberer, "Tunable privacy for access controlled data in peer-to-peer systems," in *2010 22nd International Teletraffic Congress (lTC 22).* IEEE, 2010, pp. 1–8.

[172] E. Duriakova, E. Z. Tragos, B. Smyth, N. Hurley, F. J. Peña, P. Symeonidis, J. Geraci, and A. Lawlor, "PDMFRec: a decentralised matrix factorisation

with tunable user-centric privacy," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 457–461.

[173] J. Liao, O. Kosut, L. Sankar, and F. du Pin Calmon, "Tunable measures for information leakage and applications to privacy-utility tradeoffs," *IEEE Transactions on Information Theory*, vol. 65, no. 12, pp. 8043–8066, 2019.

[174] M. K. Kundalwal, K. Chatterjee, and A. Singh, "An improved privacy preservation technique in health-cloud," *ICT Express*, vol. 5, no. 3, pp. 167–172, 2019.

[175] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM AISec*, 2019, pp. 1–11.

[176] C. Clifton, *Individually Identifiable Data*. Encyclopedia of Database Systems, Springer, 2009, pp. 1471–1472. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_1390

[177] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data mining and knowledge discovery*, vol. 2, no. 3, pp. 283–304, 1998.

[178] TensorFlow-Federated. Online: https://www.tensorflow.org/federated. Accessed 28 October 2021.

[179] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," 2019.

[180] H. B. McMahan, G. Andrew, U. Erlingsson, S. Chien, I. Mironov, N. Papernot, and P. Kairouz, "A general approach to adding differential privacy to iterative training procedures," *arXiv preprint arXiv:1812.06210*, 2018.

[181] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC CCS*, 2014, pp. 1054–1067.

[182] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *NIPS'17. arXiv:1712.07557*, 2017.

[183] T. W. Liao, "A clustering procedure for exploratory mining of vector time series," *Pattern Recognition*, vol. 40, no. 9, pp. 2550–2562, 2007. [Online]. Available: https://doi.org/10.1016/j.patcog.2007.01.005

[184] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth, “Rule discovery from time series.” in *KDD*, vol. 98, no. 1, 1998, pp. 16–22.

[185] L. Chen, Y. Gao, Z. Fang, X. Miao, C. S. Jensen, and C. Guo, “Real-time distributed co-movement pattern detection on streaming trajectories,” *Proceedings of the VLDB Endowment*, vol. 12, no. 10, pp. 1208–1220, 2019.

[186] F. Sattler, K.-R. Müller, and W. Samek, “Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints,” *arXiv preprint arXiv:1910.01991*, 2019.

[187] L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. Liu, “Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records,” *Journal of biomedical informatics*, vol. 99, p. 103291, 2019.

[188] F. Díaz González, “Federated learning for time series forecasting using LSTM networks: Exploiting similarities through clustering,” Master’s thesis, KTH, EECS, 2019.

[189] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *NIPS*, 2019, pp. 14 774–14 784.

[190] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. Quek, and H. V. Poor, “On safeguarding privacy and security in the framework of federated learning,” *IEEE Network*, 2020.

[191] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *Proceedings of the 22nd ACM SIGSAC CCS*, 2015, pp. 1310–1321.

[192] O. Choudhury, A. Gkoulalas-Divanis, T. Salonidis, I. Sylla, Y. Park, G. Hsu, and A. Das, “Differential privacy-enabled federated learning for sensitive health data,” *arXiv:1910.02578*, 2019.

[193] X. Li, Y. Li, H. Yang, L. Yang, and X.-Y. Liu, “DP-LSTM: Differential privacy-inspired LSTM for stock prediction using financial news,” *arXiv:1912.10806*, 2019.

[194] R. Tso, A. Alelaiwi, S. M. M. Rahman, M.-E. Wu, and M. S. Hossain, “Privacy-preserving data communication through secure multi-party computation in healthcare sensor cloud,” *Journal of Signal Processing Systems*, vol. 89, 10 2017.

[195] Google’s differential privacy libraries. Online: https://github.com/google/differential-privacy. Accessed 28 October 2021.

[196] Openmined/pydp: A python wrapper for google's differential privacy libraries. Online: https://github.com/OpenMined/PyDP. Accessed 28 October 2021.

[197] N. Holohan, S. Braghin, P. M. Aonghusa, and K. Levacher, "Diffprivlib: The ibm differential privacy library," 2019.

[198] Diffprivlib: The IBM Differential Privacy Library. Online: https://github.com/IBM/differential-privacy-library/. Accessed 28 October 2021.

[199] dp-stats: A library for differentially private statistics and machine learning algorithms. Online: https://gitlab.com/dp-stats/. Accessed 28 October 2021.

[200] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu, "Differentially private histogram publication," in *IEEE ICDE*, 2012, pp. 32–43.

[201] OpenDP tools. Online: https://opendp.org/. Accessed 28 October 2021.

[202] American College of Radiology. (2019) Prostate Imaging Reporting & Data System (PI-RADS). Online: https://www.acr.org/Clinical-Resources/Reporting-and-Data-Systems/PI-RADS. Accessed 28 October 2021.

[203] Plotly python graphing library. Online: https://plotly.com/python/. Accessed 28 October 2021.

[204] Y. Wang, X. Wu, and D. Hu, "Using randomized response for differential privacy preserving data collection." in *EDBT/ICDT Workshops*, vol. 1558, 2016.

[205] R. Cummings and D. Durfee, "Individual sensitivity preprocessing for data privacy," in *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020, pp. 528–547.

[206] M. Bun and T. Steinke, "Average-case averages: Private algorithms for smooth sensitivity and mean estimation," *arXiv preprint arXiv:1906.02830*, 2019.

[207] S. Tu. (2013) Introduction to Differential Privacy. Online: https://stephentu.github.io/writeups/6885-lec20-b.pdf. Accessed 28 October 2021.

Appendix A: Additional Results for Clustered Federated Learning for Privacy Preserving Time-Series Forecasting

Fitbit-GAN dataset

Preliminary results without grid search

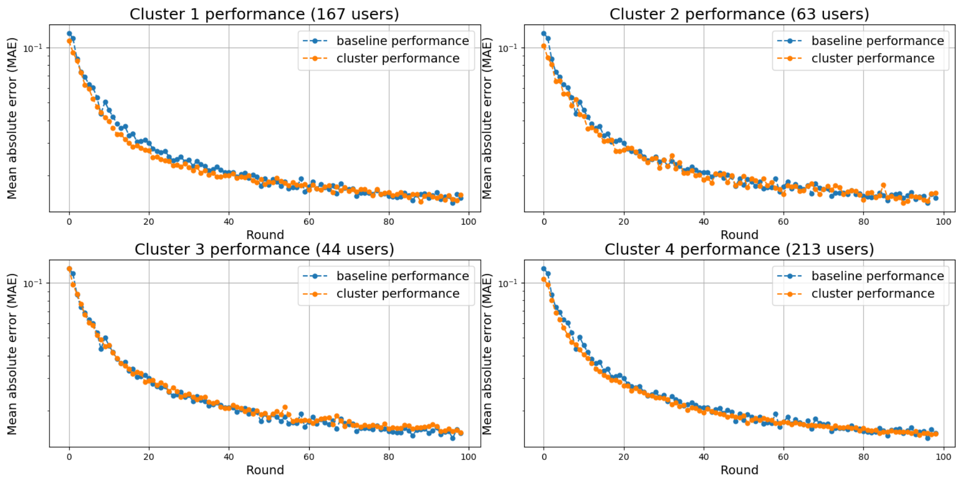


Figure 1: Evolution of the Mean Absolute Error of the baseline model against clustered model during the training process on the augmented Fitbit dataset without grid search performed on each group of users.

Table 1 presents the results obtained before running a grid search for parameters for each dataset corresponding to a cluster of users. The average change in observed error has been recorded in comparison with the model trained on the whole dataset. An increase in the average observed error suggests a decrease in prediction accuracy.

Table 1: Accuracy obtained by training one model per cluster of users on the augmented Fitbit dataset.

	Predicted	MAE	RMSE	Average change in observed error	Training time (sec)
Cluster 1 (167 users)	Macronutrients	1.001	1.277	+24%	2761
	Calories burned	14.248	18.451		
	Resting heart rate	0.063	0.079		
	Active minutes	2.476	3.176		
Cluster 2 (63 users)	Macronutrients	1.412	1.798	+50%	1219
	Calories burned	12.015	15.370		
	Resting heart rate	0.081	0.105		
	Active minutes	3.269	4.058		
Cluster 3 (44 users)	Macronutrients	1.771	2.289	+124%	1020
	Calories burned	20.327	27.603		
	Resting heart rate	0.136	0.179		
	Active minutes	4.498	5.926		
Cluster 4 (213 users)	Macronutrients	1.036	1.370	+25%	3531
	Calories burned	12.683	16.216		
	Resting heart rate	0.053	0.071		
	Active minutes	3.374	4.180		

Differential privacy - noisy data approach

Tables 2 and 3 show the results of noising the training data to achieve DP in the clustered FL scenario with different ε for noise addition. A decrease in the average observed error(compared to baseline model) implies an increase in accuracy.

Table 2: Results of noising the training data with $\varepsilon = 0.025$ noise addition to achieve DP in the clustered FL scenario

	Predicted	MAE	RMSE	Average change in observed error	Training time (sec)
Cluster 1 (112 users)	Macronutrients	0.783	0.888	-38%	294
	Calories burned	4.784	5.7		
	Resting heart rate	0.052	0.063		
	Active minutes	1.875	2.103		
Cluster 2 (57users)	Macronutrients	0.643	0.745	-40%	187
	Calories burned	6.125	7.317		
	Resting heart rate	0.054	0.064		
	Active minutes	1.79	2.067		
Cluster 3 (78 users)	Macronutrients	0.632	0.736	-42%	226
	Calories burned	6.206	7.525		
	Resting heart rate	0.049	0.058		
	Active minutes	1.668	1.973		
Cluster 4 (35 users)	Macronutrients	0.636	0.745	-36%	149
	Calories burned	8.957	10.64		
	Resting heart rate	0.052	0.062		
	Active minutes	1.769	2.093		
Cluster 5 (54 users)	Macronutrients	0.639	0.75	-34%	192
	Calories burned	10.76	12.491		
	Resting heart rate	0.049	0.059		
	Active minutes	1.734	2.059		

Table 3: Results of noising the training data with $\varepsilon = 1$ noise addition to achieve DP in the clustered FL scenario

	Predicted	MAE	RMSE	Average change in observed error	Training time (sec)
Cluster 1 (37 users)	Macronutrients	0.692	0.863	+12%	148
	Calories burned	12.359	14.447		
	Resting heart rate	0.044	0.055		
	Active minutes	3.791	4.121		
Cluster 2 (167 users)	Macronutrients	0.497	0.597	-20%	385
	Calories burned	8.687	10.393		
	Resting heart rate	0.041	0.048		
	Active minutes	1.751	2.051		
Cluster 3 (47 users)	Macronutrients	0.538	0.649	-22%	169
	Calories burned	8.752	10.598		
	Resting heart rate	0.041	0.049		
	Active minutes	1.928	2.263		
Cluster 4 (108 users)	Macronutrients	0.571	0.679	-22%	279
	Calories burned	9.235	11.176		
	Resting heart rate	0.042	0.05		
	Active minutes	1.723	2.038		