Karlsruhe Institute of Technology

Fraunhofer IOSB

Federated Learning for Private Synthetic Data Generation

MASTER'S THESIS

KIT – KARLSRUHE INSTITUTE OF TECHNOLOGY
FRAUNHOFER IOSB – FRAUNHOFER INSTITUTE OF OPTRONICS, SYSTEM TECHNOLOGIES AND IMAGE EXPLOITATION

Moritz Leitner

July 3, 2023

Reviewers:	Prof. Dr.-Ing. habil. Jürgen Beyerer
	Prof. Dr.-Ing. habil. Thomas Längle
Advisor:	M.Sc. Arno Appenzeller

Interaktive Echtzeitsysteme

This work is licensed under a Creative Commons “Attribution 4.0 International” license.

Statement of Authorship

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis in der gültigen Fassung beachtet habe.

Karlsruhe, den 3. Juli 2023

(Moritz Leitner)

Abstract

The digital transformation of the healthcare sector has gained momentum in recent years, as illustrated by the introduction of electronic health record (EHR) systems and digital infrastructures that allow the exchange of data between all stakeholders in the healthcare domain. In Germany, insured individuals will soon have the option to voluntarily donate their data stored in the nationwide EHR system for medical research purposes. While the secondary use of real-world medical data holds great potential, such as by monitoring long-term outcomes related to specific treatments, it also raises significant privacy concerns, as health data require strict protection due to the risk of stigmatization or discrimination if misused.

For this reason, various privacy-enhancing technologies (PETs) have been proposed in the literature. One such PET is differential privacy (DP), which bounds the privacy impact of data analyses by injecting carefully calibrated noise. Moreover, with recent advances in machine learning, the generation of synthetic data using generative adversarial networks (GANs) has gained attention as a privacy-preserving technique. Additionally, federated learning (FL) allows the training of machine learning models in a decentralized manner. Combining DP, synthetic data generation (SDG), and FL enables the collaborative generation of synthetic data that provide both strong privacy guarantees and value for research, while at the same time the training data do not have to be shared with a central entity.

In this master's thesis, we propose a novel approach called DP-Fed-CTGAN for generating synthetic tabular data utilizing FL under rigorous DP guarantees. Compared to existing approaches, DP-Fed-CTGAN aims to minimize the amount of information that clients have to share about their local training datasets during the FL procedure. We evaluate the performance of our open-source implementation of DP-Fed-CTGAN in terms of various utility and fidelity metrics, considering both medical and commonly used machine learning datasets. The results demonstrate that DP-Fed-CTGAN not only achieves comparable utility and improved fidelity compared to the centralized setting represented by DP-CTGAN, but can also help to increase patient acceptance of a data donation and to facilitate adherence to data protection legislation.

Zusammenfassung

Die digitale Transformation des Gesundheitswesens hat in den letzten Jahren an Dynamik gewonnen, wie die Einführung von *Electronic Health Record* (EHR)-Systemen und digitalen Infrastrukturen zum Datenaustausch zwischen allen Akteuren im Gesundheitssektor zeigt. In Deutschland werden Versicherte demnächst die Möglichkeit haben, die in ihrer elektronischen Patientenakte gespeicherten Daten freiwillig für medizinische Forschungszwecke zu spenden. Die Sekundärnutzung medizinischer Real-World-Daten birgt zwar ein großes Potenzial, etwa bei der Überwachung von Langzeitergebnissen im Zusammenhang mit bestimmten Behandlungen, wirft aber auch erhebliche Bedenken hinsichtlich des Schutzes der Privatsphäre auf, da Gesundheitsdaten aufgrund des Risikos von Stigmatisierung oder Diskriminierung infolge einer missbräuchlichen Nutzung besonders schützenswert sind.

Aus diesem Grund wurden in der Literatur verschiedene *Privacy-Enhancing Technologies* (PETs) vorgestellt. So ermöglicht beispielsweise *Differential Privacy* (DP), die Auswirkungen von Datenanalysen auf die Privatsphäre durch Einfügen von sorgfältig kalibriertem Rauschen zu begrenzen. Mit den jüngsten Fortschritten im Bereich des maschinellen Lernens hat die Generierung synthetischer Daten (SDG) mithilfe von *Generative Adversarial Networks* (GANs) als Verfahren zum Schutz der Privatsphäre an Aufmerksamkeit gewonnen. Des Weiteren erlaubt *Federated Learning* (FL) das dezentrale Training von Machine-Learning-Modellen. Durch die Kombination von DP, SDG und FL können synthetische Daten kollaborativ erzeugt werden, die sowohl starke Datenschutzgarantien als auch einen Mehrwert für die Forschung bieten, während gleichzeitig die Trainingsdaten nicht mit einer zentralen Instanz geteilt werden müssen.

In dieser Masterarbeit wird ein neuartiger Ansatz namens DP-Fed-CTGAN zur Erzeugung synthetischer tabellarischer Daten vorgestellt, der auf FL beruht und strikte DP-Garantien erfüllt. Verglichen mit bestehenden Ansätzen zielt DP-Fed-CTGAN darauf ab, die Menge an Informationen zu minimieren, die Clients während des FL-Verfahrens über ihre lokalen Trainingsdatensätze preisgeben müssen. Die Performanz der Open-Source-Implementierung von DP-Fed-CTGAN wird anhand gängiger Metriken evaluiert, wobei sowohl medizinische als auch häufig verwendete Machine-Learning-Datensätze betrachtet werden. Die Ergebnisse zeigen, dass DP-Fed-CTGAN nicht nur einen vergleichbaren Nutzen und eine verbesserte Realitätsnähe im Vergleich zum zentralen Ansatz von DP-CTGAN erreicht, sondern auch dazu beitragen kann, die Akzeptanz der Patienten für eine Datenspende zu erhöhen und die Einhaltung der Datenschutzgesetze zu erleichtern.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

k-NN k-Nearest Neighbors 60

AUPRC Area Under the Precision-Recall Curve xiii, 55, 62, 63, 68–71

AUROC Area Under the Receiver Operating Characteristic Curve xiii, 55, 62, 63, 68–70, 72

BCE Binary Cross-Entropy 18, 46, 47

BfArM Federal Institute for Drugs and Medical Devices 1, 6, 39

cGAN Conditional GAN 21, 22, 46

CTGAN Conditional Tabular GAN 2, 3, 21, 22, 24, 34–36, 39, 41, 46–48, 50, 51, 58–60, 62–64, 67, 72, 73, 75, 76

DaTraV Data Transparency Regulation 7

DP Differential Privacy v, vii, 1–3, 6, 8–16, 22, 24–27, 29, 31, 32, 34, 35, 39–41, 47, 48, 50, 58, 62, 63, 67, 71, 72, 75–77

DP-SGD Differentially Private Stochastic Gradient Descent xv, 22–24, 27, 29, 35, 45

EHR Electronic Health Record v, vii, 1, 5–7, 16, 76

ELBO Evidence Lower Bound 17

EMD Earth Mover's Distance 20, 56

ePA Elektronische Patientenakte 1, 7

ePHI Electronic Protected Health Information 6

F2A Forgiver-first Aggregation 32

F2U Forgiver-first Update 32

FedAvg Federated Averaging xv, 29, 30, 33

FedSGD Federated SGD 29

FL Federated Learning v, vii, xi, 2, 3, 6, 14, 26–32, 34–36, 39, 40, 42, 47, 51, 59, 67, 72, 75, 76

FN False Negative 54

FP False Positive 54

FPR False Positive Rate 55

GAN Generative Adversarial Network v, vii, xi, xv, 2, 14, 16–22, 24, 25, 31–35, 41, 57, 62, 63, 70–72

GDPR General Data Protection Regulation 1, 6

GMM Gaussian Mixture Model 43, 44, 59

GPU Graphics Processing Unit 16, 60, 76

gRPC GRPC Remote Procedure Calls 36

HE Homomorphic Encryption 27, 76

HIPAA Health Insurance Portability and Accountability Act 6, 7

HL7 Health Level Seven International 5

JSD Jensen-Shannon Distance xiii, 20, 56, 63, 64, 68, 69, 71, 72

KLD Kullback-Leibler Divergence 12, 17, 20, 33, 56

medGAN Medical GAN 20, 21

MLP Multilayer Perceptron 60

MPC Secure Multi-Party Computation 27, 40, 76

non-IID	Not Independent and Identically Distributed xiii, 29, 32–34, 39, 48, 60, 67, 68, 72, 75
PATE	Private Aggregation of Teacher Ensembles xi, 22, 24, 25
PDF	Probability Density Function 9, 42
PET	Privacy-enhancing Technology v, vii, 1, 2, 6, 7, 13, 15, 35, 36, 40
PMF	Probability Mass Function 22, 47, 49
PPDM	Privacy-preserving Data Mining 7, 8
PPDP	Privacy-preserving Data Publishing 7, 8
RDP	Rényi Differential Privacy 12, 13, 23, 25, 34, 35, 45, 46
SDG	Synthetic Data Generation v, vii, xi, xiii, 2, 3, 6, 13, 14, 16, 26, 28, 31, 40, 61, 63, 64, 70, 73, 75, 76
SDK	Software Development Kit 36
SDV	The Synthetic Data Vault Project 36, 50, 58
SGA	Stochastic Gradient Ascent 18, 19
SGB V	German Code of Social Law Book V 6, 7
SGD	Stochastic Gradient Descent 18, 19, 22, 23, 29
SNOMED CT	Systemized Nomenclature of Medicine - Clinical Terms 5
TEE	Trusted Execution Environment 40, 76
TN	True Negative 54, 63
TP	True Positive 54
TPR	True Positive Rate 55
TRTR	Train on Real, Test on Real 53, 60, 62
TSTR	Train on Synthetic, Test on Real 53, 60, 62

UCI University of California, Irvine 56

UML Unified Modeling Language 50

VAE Variational Autoencoder 16, 17

VGM Variational Gaussian Mixture xi, xv, 22, 34, 42–45

VM Virtual Machine 60

WD Wasserstein-1 Distance xiii, 20, 56, 64, 67–69, 71, 72

WGAN Wasserstein GAN 20, 21, 24, 47

zCDP Zero-concentrated Differential Privacy 12

1 Introduction

Digitalization is revolutionizing various sectors, and healthcare is no exception. In recent years, there has been significant progress in the digital transformation of healthcare systems worldwide. One notable example is the rollout of the so-called *Elektronische Patientenakte* (ePA) in 2021, which serves as an *electronic health record* (EHR) system for all patients covered by statutory health insurance in Germany. Building on this digital infrastructure connecting healthcare providers, insurers, and patients, beginning in 2023 insured individuals will be able to voluntarily donate their data stored in the ePA to the Health Data Lab at the *Federal Institute for Drugs and Medical Devices* (BfArM) for medical research purposes. However, the implementation details and legal aspects of this data donation scenario are still under discussion [ISH22].

While the prospect of a large data repository and thus the secondary use of real-world medical data promises enormous improvements for the treatment of diseases through the use of big data and machine learning techniques, it also raises significant privacy concerns. Health data are particularly in need of safeguarding, as misuse, for instance as a result of a cyber-attack can lead to stigmatization or discrimination of the affected individuals. For this reason, the processing of data concerning health is prohibited under the *General Data Protection Regulation* (GDPR) in the European Union unless there is an applicable exemption such as an informed consent or a transformation of the data in such a way that the individuals behind the data can no longer be identified [SK20; Mul19].

The latter can be achieved using a variety of *privacy-enhancing technologies* (PETs), which offer mechanisms to protect privacy while enabling meaningful analyses of the data. Anonymization-based approaches were widely employed to prevent the re-identification of individuals represented by the data, but proved to be vulnerable to background knowledge attacks. As a result, a mathematical measure called *differential privacy* (DP) was proposed in the computer science literature to quantify the privacy impact of algorithms analyzing the data. Mechanisms that satisfy the definition of DP usually add noise to the function being computed to ensure that the result is independent of the presence of a particular individual in the dataset [Fun+10].

However, with recent advances in machine learning, *synthetic data generation* (SDG), which was originally referred to as data augmentation to artificially expand the size of training datasets, has come into the focus of privacy research. Here, the objective is to generate synthetic data that do not allow any conclusions to be drawn about the individuals whose data were used during training, but at the same time have statistical properties that are as similar as possible to serve as a valuable asset for research. *Generative adversarial networks* (GANs) emerged as a powerful approach for generating synthetic data, where two neural networks, the generator and the discriminator, compete against each other in a zero-sum game [Goo+14]. With *conditional tabular GAN* (CTGAN), there is also a variant specifically designed for tabular data to avoid problems such as vanishing gradients and mode collapse arising from the characteristics of tabular data. Additionally, GANs can take into account the concept of DP in order to counter membership inference attacks and provide provable privacy guarantees by injecting noise to the gradients during training [Ros+20].

Another PET, *federated learning* (FL), allows the training of machine learning models in a decentralized fashion. For this, each client trains a model on its local training dataset and sends only the model parameters to a server, which sends them in aggregated form back to the clients [McM+17a]. Nevertheless, the exchange of model parameters can still compromise privacy. This is due to the fact that neural networks tend to memorize training data which can be later restored with a model inversion attack. Consequently, FL is often used together with DP [ZLH19].

In the scenario of a medical data donation, FL offers the possibility that various medical institutions participate in the training of a global CTGAN model, with a trusted third party such as the Health Data Lab handling the aggregation of the parameter updates. Thus, the patient data from each facility are solely used for local training and do not have to be shared with a central entity. This not only helps to mitigate privacy and security concerns, but is also beneficial in the case where individual facilities have insufficient data to train a CTGAN model with satisfactory data generation capability. By incorporating DP, it is ultimately possible to generate synthetic data that have significant value for research while providing mathematically provable privacy guarantees. Therefore, we investigate the potential of combining DP, SDG, and FL as an approach for a privacy-friendly medical data donation in this thesis.

1.1 Contribution

While the differentially private generation of synthetic data using FL has been investigated in the literature for images, the applicability for tabular data as often found in the healthcare domain is still largely an open question. Previous works building on the CTGAN architecture such as FDP-CTGAN [FDK22] and HT-Fed-GAN [Dua+22] were insufficiently evaluated or, in the case of Fed-TGAN [Zha+21], do not provide DP guarantees.

Therefore, in this master's thesis, we propose a novel method called DP-Fed-CTGAN for generating synthetic tabular data using FL under strict DP guarantees. In contrast to existing techniques, the main design principle of DP-Fed-CTGAN is to minimize the amount of information that clients have to disclose about their local datasets to the server, thus adhering to the aim of FL and reducing the risk of privacy breaches. For the implementation, we consider state-of-the-art frameworks for DP, SDG, and FL.

Furthermore, we thoroughly evaluate the performance of DP-Fed-CTGAN by means of various utility and fidelity metrics both for the scenario of a privacy-preserving medical data donation and for commonly used machine learning datasets. In addition, we assess the effects of FL compared to the centralized setting as well as the impact of the number of clients. We also examine the influence of the privacy budget on the quality of the generated data.

1.2 Outline

This thesis is structured as follows: In Chapter 2, we summarize legal requirements for the processing of health data and discuss the fundamental concepts of DP, SDG, and FL. In Chapter 3, we survey related work. Chapter 4 introduces our approach in detail and presents the most important aspects of the corresponding implementation. In Chapter 5, we present a comprehensive evaluation of DP-Fed-CTGAN, considering various metrics and scenarios. Finally, we conclude with Chapter 6 and provide recommendations for future work.

2 Foundations

All health data that accumulates in the course of a patient's treatment in the healthcare sector or is recorded by the insured person themselves, for example with a smartwatch, are collected digitally in a so-called *electronic health record* (EHR). Commonly, EHR data are divided into two categories: structured and unstructured data. The former can only take a certain set of numeric or categorical values. Exemplary health data that belong to this category are laboratory results, vital signs, diagnosis codes or billing data. In contrast, unstructured data, such as images or free-text clinical notes, are not represented in a standardized way, even though there are efforts to store as much data as possible in a uniform format [Sar+22].

In the context of this work, only structured health data are considered. Thus, an EHR system can be thought of as a collection of tabular datasets. A tabular dataset D consists of n records with m attributes. Such an EHR system can be implemented both via a central database and with decentralized data storage, such as using the smartphones of the patients.

In order to allow interoperability and thus a smooth exchange of information between all stakeholders in the healthcare domain, the use of standardized syntax and semantics is of particular importance for structured data. At the level of syntax, the message formats and interfaces of the organization *Health Level Seven International* (HL7) prevail, while *Systemized Nomenclature of Medicine - Clinical Terms* (SNOMED CT) is generally used to achieve uniform semantics [Sar+22].

Compared to paper-based records, EHR systems offer tremendous benefits. First, they support physicians in decision-making by consolidating all relevant information, which may also come from other healthcare providers, in one central location. They also make it easier to track the medical history of a patient and allow new medical personnel to quickly assess the condition. Furthermore, real-world clinical data represent a valuable asset for secondary use by researchers, promising in turn improved treatment of patients in the future [SK20].

At the same time, health data are particularly in need of safeguarding, as misuse such as the publication of therapy session notes in the wake of the cyberattack on Vastaamo, Finland's largest psychotherapy provider, can lead to stigmatization and even blackmail of the affected individuals [Ink21].

Therefore, we will first briefly present several important legal regulations affecting the processing of health data in Section 2.1. Subsequently in Section 2.2, with *differential privacy* (DP), *synthetic data generation* (SDG) and *federated learning* (FL), *privacy-enhancing technologies* (PETs) will be introduced, which will form the basis for developing a legally compliant and privacy-friendly donation of medical data.

2.1 Regulations Concerning the Processing of Health Data

As early as the 4th century B.C., medical confidentiality and thus the careful handling of health information was enshrined in the Hippocratic Oath. In the United States today, the regulations of the *Health Insurance Portability and Accountability Act* (HIPAA), enacted in 1996, govern the collection, storage, and sharing of *electronic protected health information* (ePHI) by physicians, hospitals, and insurers. In this context, the term ePHI includes all data that can be directly linked to an individual. The HIPAA Security Rule requires covered entities to implement administrative, physical, and technical safeguards to ensure the confidentiality, integrity, and availability of ePHI [Dro+17; Mul19; SK20]. According to § 164.514 of the HIPAA Privacy Rule, the so-called Safe Harbor method can be applied to de-identify records and then share them at will, which has led to a multi-billion dollar market centered around the monetization of EHR data. This involves removing only 18 specific features from ePHI, including name, social security number, and dates that relate to an individual [Dro+17; MP21].

In contrast, with the *General Data Protection Regulation* (GDPR), there is comprehensive data protection legislation in place in the European Union. Article 9 (1) GDPR states that data concerning health are one of the special categories of personal data whose processing is prohibited unless there is an applicable exemption, such as processing by a person bound by professional secrecy, an informed consent of the concerned subject or for research purposes after transformation of the data in such a way as to preclude the identification of the underlying individuals. On the other hand, the GDPR does not apply when working with anonymized data, as these are not personal data [Mul19; Sch+21].

In Germany, the Health Data Lab at the *Federal Institute for Drugs and Medical Devices* (BfArM) is concerned with enabling the secondary use of health data from persons covered by statutory health insurance in accordance with the *German Code of Social Law Book V* (SGB V), §§ 303a to 303f. So far, this only includes billing data, but starting in 2023 with § 363 SGB V, it will also be possible for insured individuals to voluntarily donate data stored in the national EHR system, called *Elektronische Patientenakte* (ePA), for research purposes [ISH22]. However, the concrete implementation is still vague at the time of writing, and some aspects regarding data protection are still controversial. One reason for this is that, although the data are transmitted pseudonymized to the Health Data Lab, there are no clear requirements in § 10 *Data Transparency Regulation* (DaTraV) for the anonymization methods that are used in the course of the data release to authorized users.

2.2 Privacy-Enhancing Technologies

To minimize the risk of re-identification and to comply with the various privacy laws, a multitude of *privacy-enhancing technologies* (PETs) have been proposed in the computer science literature in recent years. PETs rely on techniques such as encryption, decentralization or the distortion of (aggregated) data using privacy models, which always involve a trade-off between privacy protection and data utility. Privacy models seek to mitigate threats related to the various attribute types of tabular data that are described below [Zig+20; Sch+21]:

Explicit Identifiers: Attributes like name, mailing address, and social security number which directly identify an individual [Fun+10].

Quasi-Identifiers: A combination of attributes that, possibly using external information, allows re-identification. For instance, Sweeney [Swe02] succeeded in unambiguously identifying then-governor William Weld in a "de-identified" medical dataset from the Massachusetts Group Insurance Commission by linking it to a voter list via shared quasi-identifying attributes, specifically date of birth, ZIP code, and gender. There are many other prominent cases of such re-identification attacks, such as those described in [NS08] and [RHM19], which is why HIPAA's Safe Harbor method seems questionable.

Sensitive Attributes: Attributes that the corresponding individual does not want to be associated with, e.g., diagnosis code or salary [Zig+20].

Data evaluation can generally take place in two different settings, for which different categories of privacy models are applicable. In the first setting, *privacy-preserving data publishing* (PPDP), the data custodian publishes a sanitized version of the dataset, which the researcher can use to perform arbitrary analyses. With *privacy-preserving* *data mining* (PPDM), on the other hand, the data remains under the control of the data custodian and the researcher can only submit queries over an interface [Fun+10; Vim+12].

For many years, the research field of privacy models was dominated by the category of syntactic definitions that impose specific syntactic constraints on the anonymized dataset and are intended only for PPDP. An important approach among them is k-anonymity, where first all explicit identifiers are removed and then the dataset is transformed by generalization and suppression so that for each record there are $k-1$ others with matching quasi-identifier [Swe02]. However, the k-anonymity property provides insufficient protection for sensitive attributes in some circumstances, for example, if all k records within the group have the same value for the sensitive attribute. To address these issues, further privacy models called ℓ-diversity [Mac+07], t-closeness [LLV07], and δ-presence [NAC07] were proposed, each of which has its own strengths and weaknesses. Finally, with DP, a privacy model belonging to the semantic category was introduced and gained widespread interest as it protects by definition against attackers with arbitrary background knowledge [Fun+10].

2.2.1 Differential Privacy

In 2006, Dwork [Dwo06] presented the notion of *differential privacy* (DP), which is applicable to both PPDP and PPDM. In this publication, she first showed that a desideratum formulated by Dalenius [Dal77] in 1977, namely that an attacker with access to a database cannot learn anything about a contained individual that could not be learned without access, is impossible to achieve. Therefore, with DP, Dwork suggested a novel technique that offers a relative privacy guarantee instead of an absolute one [Vim+12]. DP is not just a theoretical construct, as it is deployed in practice by Apple [DPT17] and Google [EPK14], among others, for privacy-friendly analysis of user data.

2.2.1.1 Classical Definition

According to the definition of DP, the result of an analysis should change only slightly when a record is added to or removed from the dataset. Thus, the worst-case privacy loss for any individual is bounded regardless of their presence in the dataset, which can be formalized as follows [Fun+10; Zig+20]:

Definition 2.1 (ε-Differential Privacy) *Let $\mathcal{D}$ be the universe of all datasets. A randomized mechanism $\mathcal{M}\colon \mathcal{D} \mapsto \mathcal{T}$ gives ε-*differential privacy*, if for all datasets*

$D_1, D_2 \in \mathcal{D}$ differing on at most one record, and all $S \subseteq \mathcal{T}$ the following holds [MT07; DR13]:

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D_2) \in S].$$

A mechanism $\mathcal{M}$ satisfying this definition can be constructed by perturbing the true result of the function $f\colon \mathcal{D} \mapsto \mathbb{R}^d$ computed by the mechanism through the addition of carefully calibrated noise drawn from a probability distribution, meaning such a mechanism is non-deterministic. The privacy parameter ε controls the factor e^{ε} corresponding to the enforced upper bound on the difference of the probabilities to obtain the same specific output with two neighboring datasets, thus it quantifies the privacy loss. With a small value for ε, the allowed deviation of the probabilities is lower, leading to better privacy but also a higher amount of noise is needed affecting utility. Unfortunately, the choice of the rather abstract parameter ε for a particular use case is not trivial. The most common values found in the literature range from 0.01 to 10 [Dwo+06; DE13].

Before providing an example of such a mechanism, we must first introduce a measure, L_n-sensitivity, which determines the amount of noise that is required. This sensitivity describes the maximum change of the output caused by a single record. For example, the sensitivity of a counting query would be one. The greater the sensitivity of a function, the more noise must be added to the true result of the function [Dwo06; DR13]:

Definition 2.2 (L_n-Sensitivity) *Given two datasets $D_1, D_2 \in \mathcal{D}$ differing on at most one record, the L_n-*sensitivity *of a function $f\colon \mathcal{D} \mapsto \mathbb{R}^d$ is defined as [Dwo06]:*

$$\Delta_n f = \max_{D_1, D_2} \|f(D_1) - f(D_2)\|_n,$$

where $\|\cdot\|_n$ denotes the L_n-norm.

Among the many mechanisms for DP, the Laplace mechanism is among the most widely employed. Therefore, let us recall the *probability density function* (PDF) of the Laplace distribution with scale σ at location μ:

$$Lap(x \mid \mu, \sigma) = \frac{1}{2\sigma} e^{-\frac{|x-\mu|}{\sigma}}.$$

The Laplace mechanism simply computes the function f and adds random noise to each coordinate of the result, which is drawn from the zero-centered Laplace distribution using the L_1-sensitivity divided by ε as scale parameter. A larger ε will result in a narrower spread of the distribution, so that the true output is very likely to be only marginally distorted [Zig+20].

Definition 2.3 (Laplace Mechanism) *Given a function $f\colon \mathcal{D} \mapsto \mathbb{R}^d$, the* Laplace mechanism *$\mathcal{M}_L$ preserving ε-differential privacy is defined as [DR13]:*

$$\mathcal{M}_L(x, f(\cdot), \varepsilon) = f(x) + (Y_1, \ldots, Y_d),$$

where the Y_i are random variables drawn i.i.d. from $Lap(0, \Delta_1 f/\varepsilon)$.

2.2.1.2 Theorems and Relaxed Notions

However, there are situations for which the previously discussed definition of ε-DP would be too strict. Therefore, a relaxed variant was proposed with (ε, δ)-DP, where with probability $1 - \delta$ the original definition is fulfilled. This implies that with probability δ no privacy guarantee is given by the mechanism, so only values that are negligible in the number of records should be chosen for δ, i.e., $\delta \ll \frac{1}{|D|}$ for $D \in \mathcal{D}$ [DE13; Zig+20].

Definition 2.4 ((ε, δ)-Differential Privacy) *Let $\mathcal{D}$ be the universe of all datasets. A randomized mechanism $\mathcal{M}\colon \mathcal{D} \mapsto \mathcal{T}$ gives (ε, δ)-*differential privacy*, if for all datasets $D_1, D_2 \in \mathcal{D}$ differing on at most one record, and all $S \subseteq \mathcal{T}$ the following holds [DR13; DE13]:*

$$\Pr[\mathcal{M}(D_1) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D_2) \in S] + \delta.$$

In practice, the Gaussian mechanism is commonly used to achieve (ε, δ)-DP, where Gaussian noise scaled according to the L_2-sensitivity is added:

Definition 2.5 (Gaussian Mechanism) *Given a function $f\colon \mathcal{D} \mapsto \mathbb{R}^d$, the* Gaussian mechanism *$\mathcal{M}_G$ preserving (ε, δ)-differential privacy for any $\varepsilon < 1$ and $\sigma \geq \frac{\Delta_2 f \sqrt{2\log(1.25/\delta)}}{\varepsilon}$ is defined as [Aba+16; Mir17]:*

$$\mathcal{M}_G(x, f(\cdot), \varepsilon) = f(x) + (Y_1, \ldots, Y_d),$$

where the Y_i are random variables drawn i.i.d. from $\mathcal{N}(0, \sigma^2)$.

Two key strengths of DP stem from the post-processing and composition theorems, which are simultaneously essential for incorporating DP into machine learning algorithms. For proofs of these theorems we refer the interested reader to the monograph by Dwork and Roth [DR13]. First, the post-processing theorem states that any post-processing applied to the output of a differentially private mechanism does not compromise privacy protection, which includes noise reduction techniques and attacks based on auxiliary information.

Theorem 2.6 (Post-Processing) *Let $\mathcal{M}\colon \mathcal{D} \mapsto \mathcal{T}$ denote a (ε, δ)-differentially private mechanism. Let $g\colon \mathcal{T} \mapsto \mathcal{V}$ denote an arbitrary post-processing function. Then $g \circ \mathcal{M}\colon \mathcal{D} \mapsto \mathcal{T}$ is (ε, δ)-differentially private [DR13].*

The parallel composition theorem addresses the scenario where (possibly different) mechanisms are applied to disjoint subsets of the dataset:

Theorem 2.7 (Parallel Composition) *Let $\mathcal{M}_i$ denote a $(\varepsilon_i, \delta_i)$-differentially private mechanism for $i \in \{1, \dots, k\}$. Let X_i be disjoint subsets of the dataset D, $i \in \{1, \dots, k\}$. Then the parallel composition of the mechanisms $\mathcal{M}_i(X_i)$ provides $(\max_{i \in \{1,\dots,k\}} \varepsilon_i, \max_{i \in \{1,\dots,k\}} \delta_i)$-differential privacy [McS09].*

Since it would not be particularly useful to be limited to performing only one analysis per dataset, we now introduce the basic composition theorem regarding the sequential application of differentially private mechanisms:

Theorem 2.8 (Basic Composition) *Let $\mathcal{M}_i$ be a $(\varepsilon_i, \delta_i)$-differentially private mechanism for $i \in \{1, \dots, k\}$. Then the sequential composition of the mechanisms $\mathcal{M}_i$ satisfies at least $(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i)$-differential privacy [DR13; KOV17].*

For this reason, the parameter ε is often referred to as the privacy budget, which, after being allocated by the data custodian, can be split among several analyses by the researcher [MT07; DE13]. However, the bounds imposed by the basic composition theorem are much too loose for the iterative application of a given (ε, δ)-differentially private mechanism, as it is characteristic for the use of DP in deep learning algorithms, which will be discussed in the following sections of this thesis. In fact, at the cost of a slightly larger δ, a privacy cost ε orders of magnitude smaller can already be achieved, which is formalized in the advanced composition theorem [KOV17; JE19]:

Theorem 2.9 (Advanced Composition) *For all $\varepsilon, \tilde{\delta} > 0$, and $\delta \geq 0$, the class of (ε, δ)-differentially private mechanisms satisfies $(\tilde{\varepsilon}, k\delta + \tilde{\delta})$-differential privacy under k-fold sequential composition for [KOV17]:*

$$\tilde{\varepsilon} = \varepsilon\sqrt{2k \log(1/\tilde{\delta})} + k\varepsilon(e^{\varepsilon} - 1).$$

At the moment, there is a whole line of research known under the term privacy accounting, which aims to find and prove an even lower and more accurate privacy budget when composing differentially private mechanisms. The starting point is the Rényi divergence, a generalization of the *Kullback-Leibler divergence* (KLD), in order to quantify the difference between the probability distributions from the classical definition of DP [JE19]:

Definition 2.10 (Rényi Divergence) *Let P and Q be probability distributions over $\mathcal{T}$. For $\alpha > 1$, the* Rényi divergence of order α between P and Q *is defined as [Mir17]:*

$$D_\alpha(P \parallel Q) = \frac{1}{\alpha - 1} \log \left(\mathop{\mathbb{E}}_{x \sim Q} \left[\left(\frac{P(x)}{Q(x)} \right)^{\alpha} \right] \right).$$

Two further relaxed variants of DP, *zero-concentrated differential privacy* (zCDP) [BS16] and *Rényi differential privacy* (RDP)[Mir17], are based on the Rényi divergence. Only the latter, which bounds the divergence of the probability distributions when applying the privacy mechanism to both datasets, will be introduced in the following:

Definition 2.11 (Rényi Differential Privacy) *A randomized mechanism $\mathcal{M} \colon \mathcal{D} \mapsto \mathcal{T}$ gives* (α, ε)-Rényi differential privacy*, if for all datasets $D_1, D_2 \in \mathcal{D}$ differing on at most one record, the following holds [Mir17]:*

$$D_\alpha(\mathcal{M}(D_1) \parallel \mathcal{M}(D_2)) \leq \varepsilon.$$

In this case $\mathcal{M}$ also satisfies $(\varepsilon + \frac{log(1/\delta)}{\alpha-1}, \delta)$-differential privacy for any $0 < \delta < 1$.

With the help of RDP, it is now possible to obtain substantially tighter bounds on the true privacy costs for the sequential composition of differentially private mechanisms. Here, the informal idea is to use the equivalents of the already described mechanisms in the RDP world, keep track of the privacy budget during composition via the theorem presented below, and then convert the guarantees in terms of (ε, δ)-DP using the previously stated formula, since RDP is not as descriptive as (ε, δ)-DP is [JE19; Aso+21].

Theorem 2.12 (Composition with RDP) *The class of (α, ε)-Rényi differentially private mechanisms satisfies $(\alpha, \varepsilon k)$-Rényi differential privacy under k-fold sequential composition [Aso+21].*

Figure 2.1 visualizes the effect of each composition theorem for the same amount of noise added in each iteration. It can be seen that with repeated application of the Gaussian mechanism with noise $\sigma = 100$ at $\Delta_2 = 1$ and fixed $\delta = 10^{-6}$, considerably different results for the accumulated privacy budget ε are achieved. The sequential

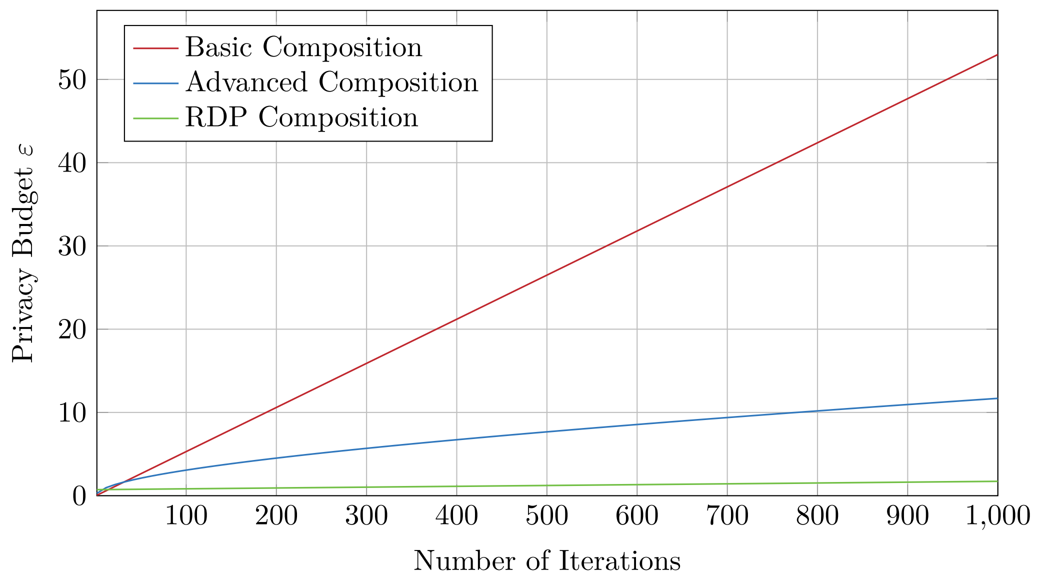


Figure 2.1: Impact of different composition theorems on the privacy budget [NA21].

composition under RDP clearly outperforms the other theorems when the mechanism is used many times. For the RDP composition Corollary 3 of [Mir17] was applied, which depicts that the Gaussian mechanism satisfies $(\alpha, (\alpha/2\sigma^2))$-RDP for a given σ if the L_2-sensitivity equals one. In the conversion from RDP to DP $\alpha = 20$ was chosen, in practice a range of values would be tested until the lowest ε was found. It should also be noted that the total δ for RDP is independent of the number of iterations, while for basic and advanced composition it grows linearly with the number of iterations. Thus, for the same amount of noise, a privacy compromise is much less likely with RDP [NA21].

2.2.2 Synthetic Data Generation

Synthetic data are artificially created to resemble real data as accurately as possible in terms of their statistical properties. At the same time, the approach of *synthetic data generation* (SDG) promises to protect the privacy of individuals better than the PETs presented so far, since the real data are only needed for the generation process and the synthetic data, as the name suggests, are not supposed to be attributable to any real human. Nonetheless, we will discuss that synthetically generated data are not free of privacy threats, but these can be addressed by combining SDG with other PETs [Fun+10; SOT22; Her+23].

While SDG can in principle be applied to all data formats, in this work we limit the scope to tabular data, for which the general idea of this approach can be formalized as follows: For a given dataset D with n entries of different individuals, the first task is to capture the information contained in the data, such as the distribution of the m attributes or the correlation between them, in some type of model (e.g., statistical or neural network model). Subsequently, the model is used to generate a synthetic dataset $\tilde{D}$ [DI21].

Two areas of application have emerged for SDG. On one hand, this includes the so-called data augmentation in the context of deep neural network training. Such networks often require huge amounts of training data, which are only available at great expense or not at all. To solve this problem, additional samples can be generated that differ in some way from the previous ones. In the case of images, this can be achieved by rotating or cropping them, for example [Jor+22; Her+23]. The second and for this thesis crucial use case is the aforementioned privacy-friendly release of synthetic data in place of real-world data, which was already proposed by Rubin [Rub93] in 1993.

In the remainder of this section, we will first discuss desirable properties of algorithms designed to generate synthetic data. Subsequently, traditional techniques that do not rely on deep learning will be briefly presented. Given the tremendous advances in machine learning in recent years, deep neural networks, most notably *generative adversarial networks* (GANs), now dominate SDG research and practice. GANs will form the main part of the elaborations since this work investigates their applicability for tabular data in conjunction with FL and DP.

2.2.2.1 Desiderata

Ideally, a synthetic data generator should fulfill the following properties, which, however, are partly in conflict with each other:

Utility: The utility of synthetic data derives from their suitability for downstream tasks and analyses for which they were generated. Since synthetic data are often used for machine learning tasks such as classification, the utility for this purpose can be assessed by comparing the performance of models trained on the real and synthetic data, respectively [Xin+22b].

Fidelity: Here, the objective is that the synthetic data mimic real data as closely as possible, which is independent of a specific application. One aspect is syntactic conformity, so that, for instance, generated ZIP codes do not suddenly contain letters. The second factor is statistical similarity, which can be assessed by comparing the distributions of attributes and relations between them [Jor+22; Her+23].

Efficiency: The runtime of the synthetic data generator should scale in a reasonable manner with the number of entries to be produced as well as with the number of attributes [Jor+22].

Privacy: If the synthesized data are too similar to the real data, this can compromise privacy, as illustrated by two types of attacks on synthetic datasets:

- **Membership Inference:** With membership inference, an attacker who has access to the synthetic data tries to determine whether a particular subject is present in the training dataset that the generator received as input. If the attacker succeeds in doing so, this is particularly problematic if the mere presence in the real dataset is stigmatizing for an individual [Xin+22b].
- **Attribute Inference:** In the context of attribute inference, the attacker can access non-sensitive attributes of the real data as well as the generated dataset. Based on this, the attacker now tries to predict the sensitive attribute values of the training dataset [Xin+22b; Her+23].

That synthetic data, like all PETs, also suffer from a trade-off between utility (or fidelity) and privacy, is underscored by a recent publication from Stadler, Oprisanu, and Troncoso [SOT22]. The authors show that state-of-the-art generative models, as introduced later, inadequately protect outliers against membership inference attacks, and that the privacy gain for outliers achieved by publishing synthetic data in place of real data is relatively small. This makes bold claims by emerging startups in this field, namely that synthetic data offer perfect privacy protection by themselves without a combination with other PETs, questionable. To counter the aforementioned threats and provide formal provable privacy guarantees, DP is often incorporated into the training process of the models later used for data generation, as elaborated in Section 2.2.3.

Although utility and fidelity are often related, it should be noted that in some situations an unchanged high utility of the data for a particular task can be achieved while reducing the fidelity in return for better efficiency or privacy [Jor+22].

2.2.2.2 Non-deep Learning Techniques

A majority of data generators that do not rely on deep learning belong to the category of approaches based on statistical modeling. Such algorithms capture the distributions of the attributes directly in a model, subsequently new entries can be synthesized by sampling from the distributions [Xin+22b]. Bayesian networks, which are graphical models of joint probability distributions, are an important representative of this category. A Bayesian network is a directed acyclic graph where the attributes correspond to the nodes and dependencies between the attributes are represented by edges. Each node is assigned a conditional probability distribution that maps each possible value of the attribute to a probability depending on the values of the parent nodes [YGP09; FV22].

The learning process of such a network consists of two steps: First, the graph structure suitable for the real data is learned, then the parameters are learned in the form of the conditional distributions. Synthetic data can now be generated by initially sampling values from the unconditional distributions of the start nodes, i.e., those that have no incoming edges. Thereafter, from the conditional probability distributions of the child nodes a value is drawn given the parent node value. This process is repeated until all attribute values are determined [Kau+20]. With PrivBayes as proposed in [Zha+14], there also exists a variant that provides DP guarantees by using noisy versions of the conditional distributions.

Simulation-based algorithms are the second type of techniques that do not involve deep learning. Such mechanisms only work for a specific domain and require expert knowledge in that area [Xin+22b]. For example, Synthea[1] is a popular open-source tool for the synthetic generation of patients and associated EHRs, which relies exclusively on publicly available health statistics and state machines for disease progression [Wal+17].

2.2.2.3 Variational Autoencoders

While the idea of deep learning, i.e., the use of neural networks with multiple layers, is not new, and LeCun et al. [LeC+89] trained a deep neural network using backpropagation as early as 1989, it took until the beginning of the last decade for deep learning to revolutionize the field of machine learning following advances in algorithms and the availability of increasingly faster *graphics processing units* (GPUs). Deep generative models, most notably *variational autoencoders* (VAEs) and *generative adversarial networks* (GANs), dominate both research and practice in the field of SDG ever since as

1 `https://github.com/synthetichealth/synthea` (visited on 01/17/2023)

well, as they often capture patterns in the data more accurately and thus provide better results [FV22; Xin+22b].

What VAEs have in common with vanilla autoencoders is that they consist of an encoder and decoder network. For an input datapoint x, the probabilistic encoder $q_\phi(z|x)$ outputs a distribution (usually a Gaussian) over the low-dimensional representations in latent space. The probabilistic decoder $p_\theta(x|z)$ in turn outputs, for a latent representation as input sampled from the distribution of the encoder, the parameters of a probability distribution over the data. For a single datapoint, the following loss function can be obtained, which is also known as the negative of the *evidence lower bound* (ELBO) [KW13; Cin+21]:

$$\mathcal{L}(x^{(i)}, \theta, \phi) = -\mathbb{E}_{q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)}|z)\right] + D_{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z)).$$

The first term corresponds to the expected reconstruction error of the datapoint with respect to the distribution of the latent representations from the encoder, while the second term acts as a regularizer that minimizes the difference between the distribution of the encoder and the prior distribution over the latent variables (generally $\mathcal{N}(0, I)$) in terms of the KLD. Once the VAE was trained by optimizing the loss with respect to the parameters ϕ of the encoder and θ of the decoder using gradient descent, synthetic data can be generated by sampling the latent variable z from $\mathcal{N}(0, I)$ and applying the decoder [KW13; WZH17].

2.2.2.4 Generative Adversarial Networks (GANs)

The second type of deep generative models are GANs, which were proposed by Goodfellow et al. [Goo+14] in 2014. Unlike VAEs, they do not explicitly estimate the distribution of the data in the form of parameters, but can generate data directly via a stochastic procedure [ML16]. GANs usually outperform VAEs and Yann LeCun, Turing Award winner for his contributions to the field of deep learning, described them as “the most interesting idea in the last 10 years in machine learning” [Gui+21]. Beyond that, GANs also are infamously known as the main technique for creating deepfakes [Ngu+19].

A GAN consists of two neural networks, the generator G and the discriminator D, which compete with each other in a zero-sum game. The generator tries to generate samples that match the real data distribution as closely as possible, while the discriminator tries to discriminate the generated samples from real samples from the training dataset as precisely as possible [Gui+21; FV22]. A frequently cited analogy is that G corresponds to a counterfeiter who would like to avoid detection of their created counterfeit money as much as possible, while D in the role of the police would like to detect the fake money [Goo+14].

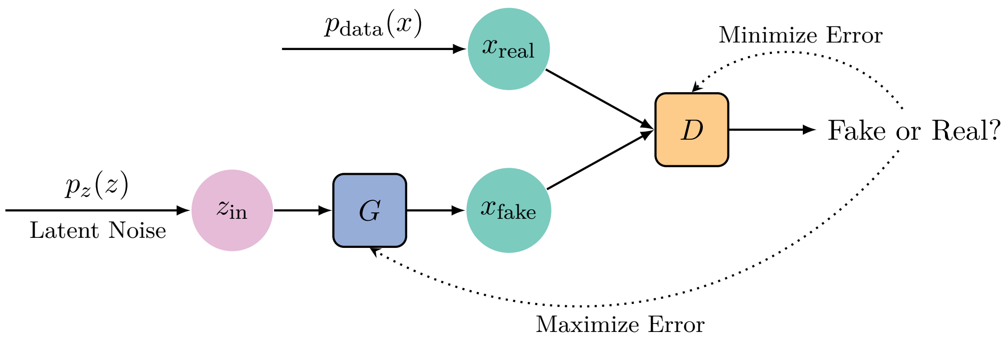


Figure 2.2: Architecture of GANs.

More formally, the generator network G learns to map noise z_{in} drawn at random from a prior probability distribution $p_z(z)$ in latent space (usually a multivariate normal distribution) to generated samples $G(z)$ in data space. The discriminator D is a binary classifier trained with synthesized datapoints and datapoints from the original dataset and outputs a scalar $D(x)$ between zero and one for a sample x, which indicates the probability that x is a datapoint from the real training dataset [Pan+19; FV22]. To classify the samples as accurately as possible, D tries to maximize the negative *binary cross-entropy* (BCE) loss between the predicted label and the actual label (real/fake), i.e., $\mathcal{L}_d = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$. In contrast, G is trained to minimize $\mathcal{L}_g = \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$ in order to fool the discriminator as effectively as possible [Pan+19; Gui+21]. The architecture of a GAN and the adversarial objectives are illustrated in Figure 2.2.

We can express the interaction between G and D as the following minimax optimization problem $V(D, G)$, where a saddle point of the loss function of the discriminator (which corresponds to the Nash equilibrium from a game theory perspective) is to be reached [Goo+14; Pan+19]:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \, .$$

Since G and D are neural networks, their respective model parameters θ_g and θ_d are optimized during training using *stochastic gradient ascent* (SGA) and *stochastic gradient descent* (SGD), respectively. The simplest procedure for training GANs is detailed in Algorithm 2.1. In each iteration, the parameters of the discriminator are updated first,

Algorithm 2.1: Training algorithm of vanilla GANs [Goo+14].

```
Input: Number of training iterations $T$, number of discriminator update steps $U$,
       learning rate $\eta$, mini-batch size $m$
Output: Parameters $\theta_g$ and $\theta_d$ of generator $G$ and discriminator $D$
1 foreach $t \in \{1, \dots, T\}$ do
2     foreach $u \in \{1, \dots, U\}$ do
3         sample a mini-batch $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_z(z)$;
4         sample a mini-batch $\{x^{(1)}, \dots, x^{(m)}\}$ from real data distribution $p_{\text{data}}(x)$;
5         $\theta_d \leftarrow \theta_d + \eta \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))\right]$;   ▷ Update
            parameters $\theta_d$ of discriminator by SGA
6     end
7     sample a mini-batch $\left\{z^{(1)}, \dots, z^{(m)}\right\}$ from noise prior $p_z(z)$;
8     $\theta_g \leftarrow \theta_g - \eta \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)})))$;   ▷ Update parameters $\theta_g$ of
        generator by SGD
9 end
```

while the parameters of the generator are fixed. This can optionally be done several times in a row until finally the model of the generator is updated [Cre+18].

Let p_g denote the distribution of samples obtained from G. Goodfellow et al. [Goo+14] showed that the globally optimal solution of the minimax game is reached when $p_g = p_{\text{data}}(x)$. They also proved that Algorithm 2.1 will converge to this solution if the neural networks have sufficient capacity and in each iteration the discriminator is optimal for a given generator G, meaning

$$D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} = \frac{1}{2}.$$

Unfortunately, during the training of vanilla GANs, as they were just introduced, some difficulties can arise, of which we will summarize the most common ones in the following:

Mode Collapse: The problem of mode collapse occurs when, as a consequence of poor generalization, the generator produces only very similar samples or, in the worst case, always the same sample, which maximally fool the discriminator. In such a situation, the training may get stuck in an impasse and the generator can only synthesize a severely limited set of distinct datapoints [Goo+14; SC21].

Vanishing Gradient: GANs are also not spared from the vanishing gradient problem that is common when training deep neural networks with backpropagation. If the samples generated by G are very unrealistic in the beginning and D can easily differentiate them from real samples, $D(G(z))$ can be almost zero. In such a case, $\log(1 - D(G(z)))$ may be very small, so the gradient may contain too little information for the model of G to improve [Cre+18; SC21].

Instability: Gradient-based methods have only limited suitability for finding the Nash equilibrium of a minimax game, which is why poor convergence may occur due to oscillating parameters [SC21; FV22].

2.2.2.5 Variants of GANs

In the literature, a multitude of different variants of GANs were presented that address specific training challenges or are optimized for particular use cases. Subsequently, we outline a selection of GAN variants which we consider to be relevant for the scope of this thesis. For further variants, we refer interested readers to the survey articles by Saxena and Cao [SC21] and Figueira and Vaz [FV22].

Wasserstein GAN (WGAN): In their publication, Arjovsky, Chintala, and Bottou [ACB17] first theoretically investigated the applicability of different statistical divergences, namely the KLD, the *Jensen-Shannon distance* (JSD), and the *Wasserstein-1 distance* (WD), which is also known as *earth mover's distance* (EMD), for gradient-based learning of distributions. They concluded that the EMD is the most suitable for this purpose because, unlike the other distances, it is continuous everywhere and differentiable almost everywhere. Mathematically, it can be shown that for an optimal discriminator, the generator of a vanilla GAN minimizes the JSD between p_{data} and p_g. Therefore, the authors proposed WGAN, where the generator instead tries to minimize the EMD [Goo+14; SC21]. With the help of the Kantorovich-Rubinstein duality, a computationally efficient objective function can finally be derived:

$$\min_{G} \max_{D \in \mathcal{D}} V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[D(x) \right] - \mathbb{E}_{z \sim p_z(z)} \left[D(G(z)) \right],$$

where D (now called critic) is an element of the set of 1-Lipschitz functions $\mathcal{D}$. With this approach mode collapse and vanishing gradients can be avoided [ACB17; Gul+17].

Medical GAN (medGAN): Vanilla GANs face difficulties when generating discrete data because the gradients during training are zero almost everywhere [BLC19]. In order to synthesize discrete variables representing medical events such as diagnoses or medications, Choi et al. [Cho+17] introduced medGAN, which inserts a decoder of a pre-trained autoencoder between the generator and the discriminator. This decoder converts the continuous output of the generator into a discrete sample. The autoencoder is trained with the real data and can be viewed as an additional pre-trained hidden layer in the generator during actual GAN training. Additionally, the authors use mini-batch averaging, i.e., the average of the samples in the mini-batch is passed to the discriminator to prevent mode collapse.

Conditional GAN (cGAN): Vanilla GANs, after being trained with a labeled dataset, do not provide a way to generate synthetic datapoints of a certain class, since the latent space cannot be interpreted and thus it is not known from which range in the latent space to sample from in order to obtain data of a particular class [FV22]. For example, in a medical use case, one might be interested in synthesizing patients who do and do not suffer from a specific disease, and then identifying differences in features to infer possible risk factors.

For this reason, Mirza and Osindero [MO14] proposed the cGAN architecture, in which the generator and discriminator are given an additional conditional variable c as input, which can be used to provide additional information to the models during training or subsequently to select the class of the samples to be generated. This results in the following objective function [Pan+19]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x|c)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1 - D(G(z|c)))\right].$$

Conditional Tabular GAN (CTGAN): With CTGAN, Xu et al. [Xu+19] introduced an architecture that is based on a further improved variant of WGAN, but makes additional adjustments to address the characteristics of tabular data. For example, attributes with continuous values are often not Gaussian distributed, which can cause the gradients to vanish when the common min-max normalization is used. Furthermore, continuous columns often have multiple modes, of which not all are captured by the model, especially in the case of vanilla GANs. In addition, the values (or categories represented by them) of discrete columns are often imbalanced, which causes mode collapse or inadequate learning of minority classes [Xu20; FV22]. The authors employ the following techniques to overcome these issues:

- **Mode-specific Normalization:** To represent continuous attributes, the authors developed a mode-specific normalization that first estimates the number of modes using a *variational Gaussian mixture* (VGM) model. Then, each value is represented by a one-hot vector that specifies the mode and a scalar that indicates the value within the mode [Xu+19].

- **Conditional Generator:** In order for CTGAN to better cope with imbalanced discrete columns, a conditional generator like in the cGAN architecture is used. For this, the conditional vector c represents the categories for all discrete columns in one-hot encoded form. Furthermore, a technique called training-by-sampling is used that allows the generator to learn the conditional distribution of the real data. Here, a discrete column is first chosen at random. Then a random value is drawn based on a *probability mass function* (PMF), where the probability mass for each category of the selected column corresponds to the logarithm of the frequency. Finally, this condition is expressed as a conditional vector and used as input for the generator [FV22].

2.2.3 Differentially Private Deep Learning

As mentioned earlier, even with synthetic data, there are threats to the privacy of individuals in the training dataset. However, for neural networks in general, there is also the risk that a model overfits on certain training data and stores them unintentionally so that they can be restored with a model inversion attack as shown in [FJR15]. For this reason, DP is increasingly being incorporated into the training process to provide mathematically rigorous privacy guarantees. In this regard, one can integrate DP into machine learning algorithms in three ways. With objective perturbation, the first possibility is to add noise to the objective function $J(\theta)$, which is optimized by the algorithm. Second, one can add noise to the gradient $\nabla_\theta J(\theta)$ in each iteration, which is called gradient perturbation. Finally, noise can also be added to θ^*, the output of the algorithm, using output perturbation [JE19]. In the following, we present *differentially private stochastic gradient descent* (DP-SGD) and *private aggregation of teacher ensembles* (PATE), the two most widely employed mechanisms for differentially private training of deep neural networks, as well as their application in GANs.

2.2.3.1 Differentially Private Stochastic Gradient Descent

When training neural networks, the loss function is often minimized using mini-batch *stochastic gradient descent* (SGD) by iteratively approximating the gradient based on the samples in the mini-batch, to then update the parameters of the model by taking a step in the opposite direction of the gradient. Abadi et al. [Aba+16] proposed DP-SGD, a differentially private version of SGD based on gradient perturbation with Gaussian noise shown in Algorithm 2.2.

Algorithm 2.2: DP-SGD [Aba+16].

```
Input: Samples $\{x^{(1)}, \dots, x^{(N)}\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\theta, x^{(i)})$, number
       of training iterations $T$, learning rate $\eta_t$, mini-batch size $m$, gradient
       norm bound $C$, noise scale $\sigma$
Output: Parameters $\theta_T$ and privacy cost $(\varepsilon, \delta)$ using moments accountant
1  initialize $\theta_0$ randomly;
2  foreach $t \in \{1, \dots, T\}$ do
3      $B_t \leftarrow$ (sample a mini-batch $\{x^{(1)}, \dots, x^{(m)}\}$ randomly);
4      foreach $x^{(i)} \in B_t$ do
5          $g_t(x^{(i)}) \leftarrow \nabla_{\theta_{t-1}}\mathcal{L}(\theta_{t-1}, x^{(i)})$;          ▷ Compute gradient
6          $\hat{g}_t \leftarrow g_t(x^{(i)}) / \max\left(1, \frac{\|g_t(x^{(i)})\|_2}{C}\right)$;          ▷ Clip gradient
7      end
8      $\bar{g}_t \leftarrow \frac{1}{m}\left(\sum_{i=1}^{m}\hat{g}_t(x^{(i)}) + \mathcal{N}(0, \sigma^2 C^2 I)\right)$;          ▷ Add noise
9      $\theta_t \leftarrow \theta_{t-1} - \eta_t \bar{g}_t$;          ▷ Descent
10 end
```

What remains unchanged compared to conventional mini-batch SGD is that in each training step first the gradient of the loss function with respect to the weights is computed for each sample in the mini-batch. As already introduced in Definition 2.5, for the Gaussian mechanism the amount of noise depends on the L_2-sensitivity of the function being computed. Since the maximum change to the gradient in terms of the L_2-norm by changing a single sample is neither known in advance nor even computable, the authors use gradient clipping in DP-SGD to ensure that the L_2-norm is bounded by the parameter C. In detail, the gradient vector g is scaled to be of norm C if its norm exceeds the clipping threshold C. It should be noted that gradient clipping is part of other deep learning algorithms regardless of privacy reasons [CWH20]. With that, Gaussian noise can be added to the average of the gradients, thus masking the influence of each sample. Lastly, with the actual descent, the update of the model parameters is performed.

The moments accountant of DP-SGD, whose theoretical foundation is the composition of RDP as presented in Theorem 2.12, accumulates the privacy cost during the training process. For a given sampling probability $q = \frac{m}{N}$ and $\sigma = \frac{1}{\varepsilon}\sqrt{2\log\frac{1.25}{\delta}}$, Abadi et al. prove that Algorithm 2.2 is $(\mathcal{O}(q\varepsilon\sqrt{T}), \delta)$-differentially private for a appropriately chosen clipping threshold C [Aba+16].

There are several approaches to combine the DP-SGD mechanism with the GAN framework. Xie et al. [Xie+18] proposed DPGAN, which adapts DP-SGD for the training procedure of WGANs. The authors clip the weights of the discriminator network to fulfill the 1-Lipschitz condition and show that this simultaneously bounds the gradients so that they do not have to be clipped additionally. By applying Theorem 2.6 it follows that for a differentially private discriminator also the generator guarantees (ε, δ)-DP [Fan20]. By contrast, DP-CTGAN, which is built on the CTGAN architecture, clips the gradient and adds noise to it [Ros+20].

2.2.3.2 Private Aggregation of Teacher Ensembles

PATE is a type of ensemble learning proposed by Papernot et al. [Pap+17] that provides privacy protection based on the fact that the training consists of two steps, where access to the sensitive training data only occurs in the phase to which adversaries have no access.

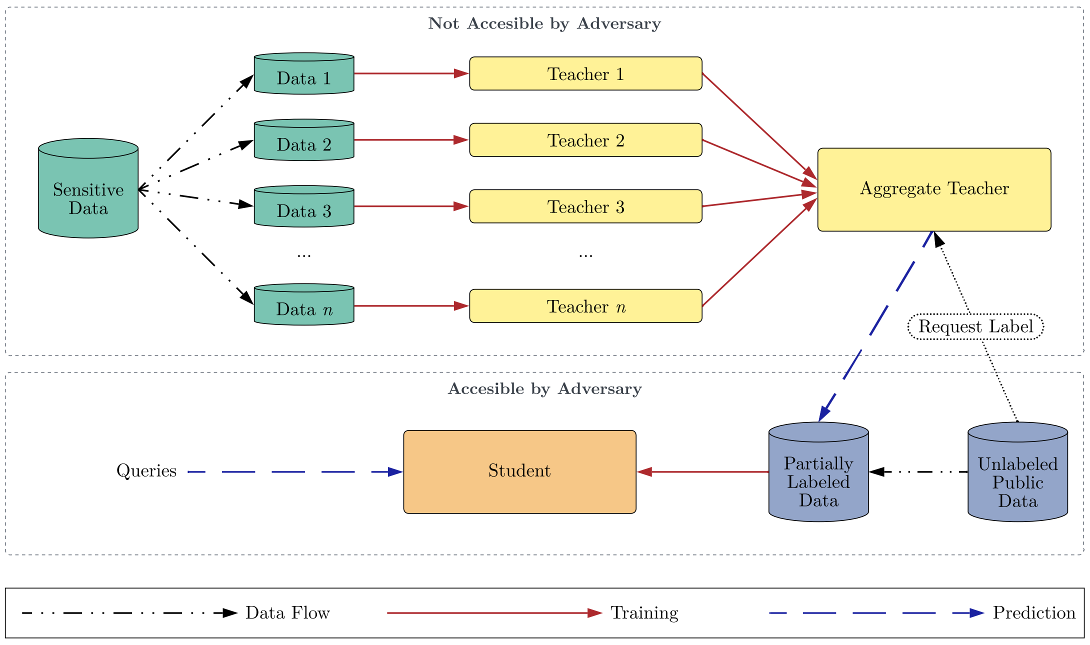


Figure 2.3: Overview of PATE [Pap+17].

As shown in Figure 2.3, the first step is to train an ensemble of n classifiers, called teachers, on disjoint subsets of the training dataset. For an unseen sample x, the predictions of the teachers are aggregated by noisy majority vote so that the decision cannot depend on a single teacher (which could leak private information about the training data of the teachers). Hence, PATE belongs to the category of differentially private algorithms that use output perturbation [Pap+17; Asl+23]. Formally, we obtain the following for the output of the aggregate teacher $f(x)$, given $n_j(x)$ denotes the number of teachers who output class j for input x [JYS19]:

$$f(x) = \operatorname*{argmax}_j \left\{ n_j(x) + Lap\left(0, \frac{1}{\varepsilon}\right) \right\}$$

However, it would not be a good idea to actually use f for inference, since with more queries the noise has to be increased to get meaningful DP guarantees, so the output would probably be useless in a real-world setting. For this reason, with the student in the second phase, another model is trained, which is deployed as a proxy of the teacher ensemble, so that the privacy loss is independent of the number of queries performed by the end user. The student is trained on publicly available unlabeled data, requesting the noisy labels of the aggregation mechanism for a part of the data. Papernot et al. show via RDP accounting that the PATE mechanism for T label queries satisfies $(4T\varepsilon^2 + 2\varepsilon\sqrt{2T \log \frac{1}{\delta}}, \delta)$-DP even when the parameters of the student are public [Pap+17]. In [Pap+18], an improved version is presented that achieves the same accuracy with lower privacy costs using the Gaussian mechanism, but is not further discussed here.

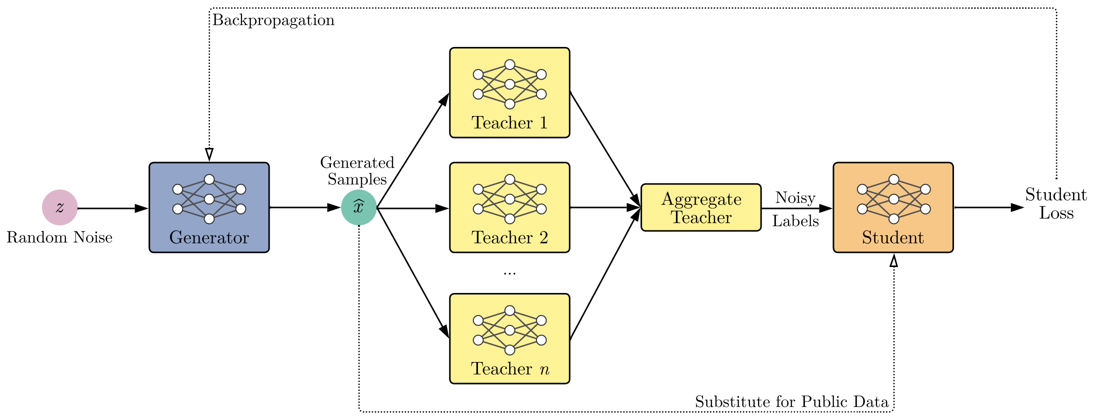


Figure 2.4: Block diagram of PATE-GAN [JYS19].

PATE-GAN adapts the PATE framework for GANs by replacing the discriminator with the PATE mechanism. Since the aggregated output of the teachers is not differentiable, the student is used for backpropagation to the generator as also illustrated in Figure 2.4. Each teacher is trained like the discriminator in a vanilla GAN, except that each teacher only sees a subset of the training dataset. Moreover, the student is trained only on the outputs of the generator, because public data for training the student would be an unrealistic assumption in the context of SDG. Also in this case, Theorem 2.6 yields the DP guarantees of the generator since this model was trained exclusively with the differentially private student [JYS19; Fan20].

2.2.4 Federated Learning

FL is an approach for privacy-preserving decentralized machine learning proposed by McMahan et al. [McM+17a] that addresses two challenges. First, in many economic sectors, data resides in isolated data islands, such as in the healthcare sector in individual hospitals. Single data islands often do not permit the training of a machine learning model with satisfactory accuracy due to limited data. On the other hand, data protection regulations impose restrictions or even bans on collecting data and sharing them with external parties. The idea of a FL system is therefore to allow K parties (or also referred to as participants or clients) $\{P_1, \ldots, P_K\}$ to collaboratively train a model M_{FED} with their respective datasets $\{D_1, \ldots, D_K\}$, without requiring any party P_k to share their dataset D_k with another party. Furthermore, the accuracy of M_{FED} should be as close as possible to the accuracy of a conventional model M_{CON} centrally trained with $\tilde{D} = D_1 \cup \ldots \cup D_K$ [Yan+19].

In the following, we will first categorize FL systems based on typical building blocks, then discuss an architecture in more detail that is relevant for the later implemented method, and briefly present exemplary algorithms.

2.2.4.1 Taxonomy

FL systems are usually categorized according to the following four aspects:

Data Distribution: With respect to the distribution of sample and feature space across the different parties, we can distinguish between horizontal, vertical, and hybrid FL. The horizontal scenario is the most common, where the dataset of each party has the same features, but there is little intersection in the sample space. In this case, a global model can be obtained by aggregating the local models in a privacy-preserving manner. In vertical FL, the parties have an overlapping sample space but diverging feature spaces, i.e., the individuals behind the samples occur in datasets of multiple parties, but each party holds different information. Thus, an additional technique is needed here to align the samples belonging to the same individual. Last, hybrid FL refers to the case where the datasets differ in both feature and sample space. There are special approaches for this setting, which are subsumed under the term federated transfer learning [Yan+19; Li+21].

Privacy Mechanism: Although the local training datasets are not shared in FL, even the parameter updates that are exchanged can compromise privacy, as it was demonstrated for the sharing of gradients in [ZLH19]. One solution to this issue is *secure multi-party computation* (MPC), which originates from cryptography research. MPC allows the participants to jointly compute a function while the inputs can remain private. This can be achieved by *homomorphic encryption* (HE) schemes, where the parties encrypt their inputs and then certain mathematical operations can be directly performed on the ciphertexts. However, MPC does not protect the global model against the aforementioned model inversion attacks, and MPC comes with a high computational overhead. Therefore, an alternative is to inject differentially private noise in the learning process [Li+21; Kai+21]. In this context, there are two different alternatives for DP guarantees, depending on the step in which the noise is added:

- **Record-level DP:** With record-level DP, each party perturbs the parameters locally, e.g., with DP-SGD, before the updates are sent to the server for aggregation [Kai+21; NHC22].
- **Client-level DP:** In contrast, if the noise is added to the aggregated updates by the server, one obtains client-level DP. To this end, the definition of DP is adapted in such a way that the datasets D_1 and D_2 differ by all records of a single participant, i.e., the influence of a participant on the final model is bounded. A disadvantage of this variant is that it requires a higher degree of trust in the server. Furthermore, important aspects of client-level DP are not yet well understood, notably when the number of samples varies strongly between participants [McM+17b; NHC22].

Unfortunately, the privacy properties of both alternatives can hardly be compared with each other, although there is a theoretical possibility for conversion with group privacy, which however is not practicable yet [NHC22].

Communication: A further distinction is possible with regard to the communication architecture. In centralized communication, a server receives the updates of the local models from the parties and aggregates them into a global model, which is sent back to the parties. With a decentralized architecture, there is no need for a server, as the parties communicate directly among themselves. This setting is also called peer-to-peer learning [Kai+21].

Scale of Federation: Under the aspect of the scale of the federation, a classification into cross-silo and cross-device FL systems can be made. Cross-silo FL is characterized by a relatively small number of parties (often corresponding to individual organizations) that have relatively large datasets and significant computing power. By contrast, in the cross-device scenario, the number of parties (which could be edge devices, for example) is high, but each party has only limited data, limited communication bandwidth, and low computing power. Here it is also assumed that the participants are unreliable and thus may not contribute in every round, for example because the battery is empty or the network connection was lost [Li+21].

2.2.4.2 Architecture of Horizontal Federated Learning

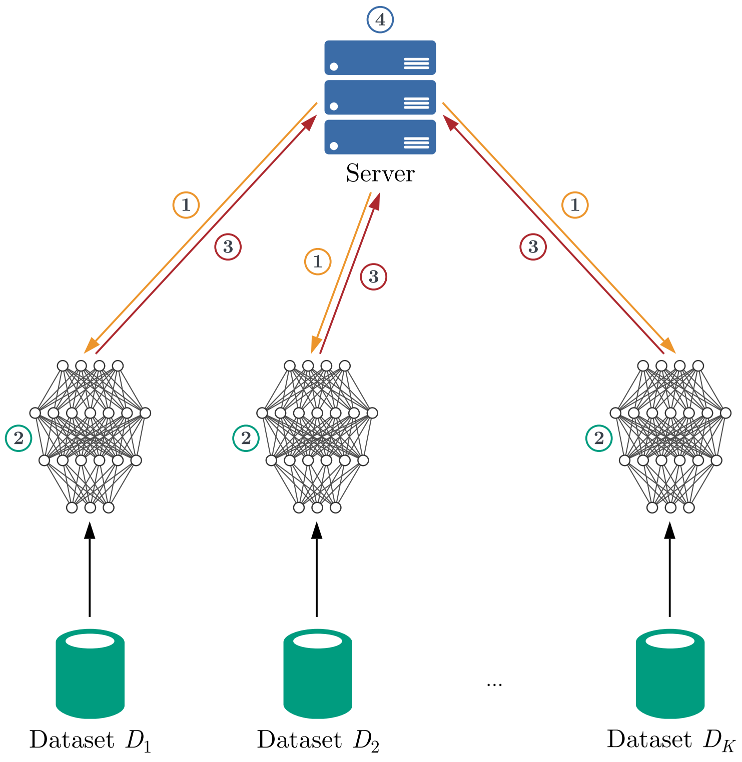


Figure 2.5: Architecture of a horizontal FL system [Yan+19].

We now explain the architecture of a horizontal FL system in more detail, since this is also the setting for the SDG method implemented later on. As can also be seen in Figure 2.5, in such a system K participants want to train a global model by exchanging information through an honest-but-curious server [Yan+19]. In the first step, the latest model parameters are sent to the participants selected for the current round. Subsequently, these participants compute an update to the model for instance by performing SGD on their local dataset. In the third step, the participants send the local updates to the server, which aggregates them using one of the privacy-preserving mechanisms presented earlier. Lastly, the server updates the global model using the result of the aggregation step [Kai+21].

2.2.4.3 Algorithms

While there are hundreds of different algorithms in the literature for all imaginable manifestations of FL systems, we want to present two simple algorithms hereafter, which, however, can be adapted for most scenarios.

The baseline algorithm for training a model in the federated setting is *federated SGD* (FedSGD). In step t, the server randomly selects a fraction C of all participants, where the selected ones are denoted by the index set S_t. Now each participant P_k with $k \in S_t$ computes the average gradient g_k of the loss function on the local dataset D_k with respect to the current model parameters θ_t. At last, the server aggregates the gradients and updates the parameters depending on the learning rate η, i.e., $\theta_{t+1} \leftarrow \theta_t - \eta \sum_{k \in S_t} \frac{|D_k|}{N} g_k$, where the weight is usually calculated by the number of samples of the participant $|D_k|$ relative to the total number of samples N. Hereafter begins the next round [McM+17a].

Federated averaging (FedAvg), which is shown in Algorithm 2.3, aims to reduce the communication overhead by allowing each selected participant to execute multiple gradient descent steps in each round. Two additional parameters are introduced for this purpose: E determines the number of local epochs per round and m the mini-batch size. It should be noted that with $E = 1$ and $m = \infty$ FedSGD is obtained. The computations that are performed on each participant are highlighted in blue in the pseudocode. Furthermore, one can simply replace the participant update step with DP-SGD in order to provide privacy guarantees with the notion of DP [McM+17a; Li+21].

However, FedAvg converges much slower than FedSGD on *not independent and identically distributed* (non-IID) data, i.e., the distribution of the attributes or labels varies greatly between the participants. This is because the larger the number of local epochs, the more the local parameters will diverge in the case of heterogeneity in the local distributions, and therefore the aggregated model will also deviate to a larger

Algorithm 2.3: FedAvg [McM+17a].

Input: Number of participants K, number of federated rounds T, fraction of participants per round C, number of local epochs E, learning rate η, mini-batch size m

Output: Parameters θ_T of global model

```
1  initialize $\theta_0$ randomly;
2  foreach round $t \in \{1, \dots, T\}$ do
3      sample subset $S_t$ of $\max(C \cdot K, 1)$ participants randomly;
4      foreach participant $k \in S_t$ in parallel do
5          initialize $\theta_t^k \leftarrow \theta_{t-1}$;
6          $\mathcal{B} \leftarrow$ (split $D_k$ into mini-batches of size $m$);
7          foreach local epoch $e \in \{1, \dots, E\}$ do
8              foreach mini-batch $B \in \mathcal{B}$ do
9                  $\theta_t^k \leftarrow \theta_t^k - \eta \nabla \mathcal{L}(\theta_t^k, B)$;
10             end
11         end
12         return $\theta_t^k$ to server;
13     end
14     $\theta_t \leftarrow \sum_{k \in S_t} \frac{|D_k|}{N} \theta_t^k$;
15 end
```

extent from the ideal model that would be obtained for IID data. FL thus requires a trade-off between communication costs and convergence speed [Zhu+21; Kai+21].

3 Related Work

After introducing the foundations for a deeper understanding of the subject area of this thesis, we now discuss related work in the field of privacy-friendly SDG using FL. To this end, the next section first summarizes the results of our literature review and highlights the contributions of this work. Subsequently, open-source libraries and state-of-the-art frameworks for DP, SDG, and FL are presented, as well as their applicability for the method implemented in the next chapter and thus the scenario of a medical tabular data donation is assessed.

3.1 Synthetic Data Generation Using Federated Learning

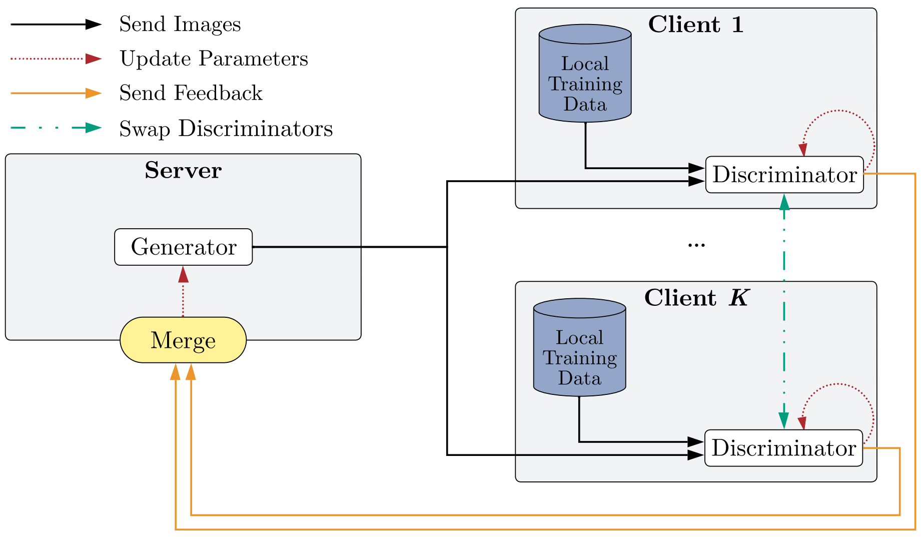


Figure 3.1: Architecture of MD-GAN [HMS19].

The MD-GAN architecture, which was introduced by Hardy, Merrer, and Sericola [HMS19] in 2019, is the first that is specifically tailored for training GANs in a federated setting, although it is intended for generating images. As shown in Figure 3.1, the generator is trained on the central server, while the discriminators are distributed among the participants of the FL system. In each round, the generator first sends every client two batches of generated samples. The first batch is used to train the respective discriminator so that it discriminates the generated samples from the real local samples. On the other hand, the second batch is used to compute the error feedback for the generator using the judges of the discriminator. Subsequently, the feedback from the discriminators is aggregated by averaging to update the parameters of the generator with the Adam optimizer. To prevent the discriminators from overfitting on their local dataset, the parameters are swapped among the discriminators after a specified number of epochs. One advantage of this approach is that a central generator reduces the computational cost for the clients. However, this is at the same time accompanied by higher communication costs due to sending the generated samples to the clients, so that the central generator eventually becomes a bottleneck if the number of clients is high. The main drawback of MD-GAN is that the authors assume that the local datasets are IID and no convergence guarantees can be obtained in the more realistic case of non-IID.

Given this, Yonetani et al. [Yon+19] proposed two adapted versions called *forgiver-first update* (F2U) and *forgiver-first aggregation* (F2A) for the situation where the classes such as different diseases are not identically distributed across the clients. The idea of F2U is to update the generator for each generated sample using only the judgment of the most forgiving discriminator, i.e., the discriminator who rated the sample as the most real. While the authors prove that this approach yields an optimal generator in terms of the class distribution, the convergence is rather slow since only a fraction of the available information is used in each training iteration. For this reason, it is suggested in the publication to use F2A in practice, and thus a weighted average of the judgments of all discriminators, with the more forgiving ones being weighted more strongly using an adaptable softmax function.

In order to provide provable privacy guarantees by means of DP against threats such as unintended memorization also for the federated training of GANs, with DP-FedAvg-GAN, Augenstein et al. [Aug+20] suggested a similar architecture to MD-GAN. In this approach, both the generator and the discriminator are located on the server, and the parameters of the discriminator are derived by training local discriminators on a subset of clients and then averaging the parameter updates. In each round, Gaussian noise is added to these aggregated parameters so that client-level DP is achieved.

To avoid the high communication overhead caused by distributing the generated samples and reporting back the error feedback or parameter updates in the techniques considered so far, the FedGAN architecture was introduced by Rasouli, Sun, and Rajagopal [RSR20] as shown in Figure 3.2. Here, the clients each train a generator and

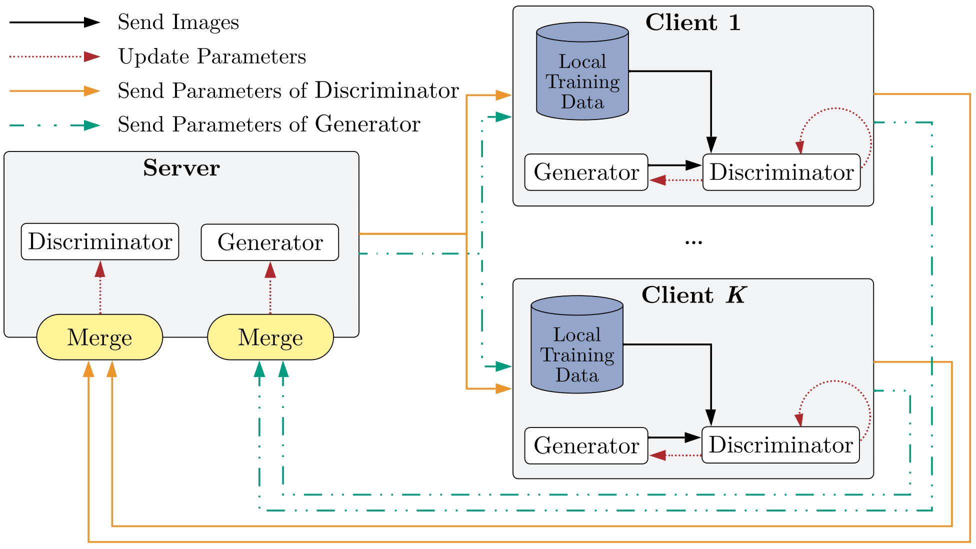


Figure 3.2: Architecture of FedGAN [HMS19].

discriminator locally and synchronize the parameters after a fixed number of training steps via an intermediary using FedAvg. Furthermore, the authors prove that the FedGAN algorithm converges even for non-IID data. It should be noted that the clients in return have a higher computational load in this scenario.

Li et al. [Li+22] proposed a variant, IFL-GAN, where only the parameters of the local generators are aggregated by weighting the updates using the maximum mean discrepancy between the local training data distribution and the generated data distribution normalized with the softmax function. The authors demonstrate that this improves convergence by preventing local models that have already reached the Nash equilibrium from jumping out of it.

FeGAN [Gue+20], where each client also has its own generator and discriminator, seeks to accelerate the convergence of FedGAN and mitigate common training issues of GANs. The authors achieve the former by requiring each client to submit metadata related to the local dataset, such as the number of classes and the number of samples per class, to the server prior to the actual training. Then, the server uses this metadata to calculate the KLD between the local dataset and the global data distribution and applies the negative score to weight the client updates using the softmax function so that clients with a large discrepancy have a smaller impact on the result of the aggregation. Yet, preprocessing of the local datasets is required to extract the metadata, for example using k-means clustering. To overcome the common problem of mode collapse when training GANs, FeGAN relies on balanced sampling, i.e., in each round, when selecting the contributing clients, preference is given to those that have many samples or a similar number of samples for each class. In addition, they do not limit the evaluation of FeGAN to high-performance server hardware like the publications reviewed so far, but also consider devices with limited memory or processing power that only slightly degrade convergence. While the authors show that this approach actually allows them to learn the data distribution better in less time than in the centralized case without FL or when compared to MD-GAN, the sharing of metadata requires greater trust in the server.

Xin et al. [Xin+20] designed the private FL-GAN architecture, which was improved in [Xin+22a] for non-IID data. After initialization on the server, the generator and discriminator parameters are updated sequentially by each client to minimize the number of accesses to each local dataset and thus information leakage. Furthermore, when training the discriminator, Gaussian noise is added to the gradients, resulting in record-level DP through Theorem 2.7. The authors thereby employ RDP accounting to keep track of the privacy budget. In order to provide adequate results even in the situation of non-IID, private FL-GAN relies on lifelong learning, i.e., the local dataset of each client is augmented with samples obtained from the generator based on the parameters of the previous client and an additional regularization term is introduced in the loss function of the discriminator to ensure that the model does not “forget” the data distribution of the previous client. It should be noted that this does not affect the DP guarantees because of Theorem 2.6. Although this approach is communication efficient, the training time is increased due to the sequential updates.

All approaches described so far use GAN variants that have been specifically optimized for images, or have been applied and evaluated only on image datasets such as MNIST or CIFAR-10. However, with the differentially private generation of (medical) tabular data in a federated setting, which is the investigated use case of this thesis, specific challenges already discussed in the last chapter arise such as even “faster” vanishing gradients and difficulties with multimodal distributed continuous columns and imbalanced discrete columns. To the best of our knowledge, this scenario has rarely been investigated in the literature so far.

Fed-TGAN [Zha+21] trains a CTGAN model in a federated manner using local generators and discriminators. Since CTGAN represents each column as a one-hot vector and requires mode-specific normalization for continuous attributes, the clients must agree on a uniform encoding. For this purpose, the clients send the frequencies of the categories for discrete columns and the fitted VGM models for continuous attributes to the server, which determines the encoding scheme from this information and sends it back to the clients. To improve convergence, a divergence matrix is calculated based on the information shared with the server, which determines the weight for each client depending on the columns when aggregating the local models. However, DP is not integrated into the training process of Fed-TGAN. Further, the capabilities of the generator are only evaluated for a fixed number of five clients and the authors have not made the code publicly available.

An almost identical approach is taken in HT-Fed-GAN [Dua+22], where additionally Gaussian noise is added to the aggregated parameters of the discriminator to satisfy client-level DP. This variant is evaluated in the publication only for three clients with a fixed $\varepsilon = 0.5$ and a questionably chosen $\delta = 9.8$. Moreover, the associated code was not published.

Fang, Dhami, and Kersting [FDK22] proposed FDP-CTGAN, a differentially private version of CTGAN adapted for FL. In this architecture, Gaussian noise is injected to the local gradients of the discriminators, so that record-level DP is guaranteed. The performance of FDP-CTGAN is competitive with the centralized setting on multiple medical datasets for fixed privacy parameters. Unfortunately, according to the implementation[1] on GitHub, the authors train three clients on the same local dataset, which contradicts the aim of FL.

In summary, it is evident that the considered use case requires a more detailed analysis and, in particular, a comprehensive evaluation, such as the influence of the number of participants and the impact of different privacy budgets, which will be addressed in the following chapters of this thesis.

3.2 Frameworks for Privacy-Preserving Federated Synthetic Data Generation

Before we move on to the implementation of our method for a privacy-friendly donation of medical data using GANs, DP, and FL, we first discuss a selection of existing libraries and frameworks that provide these PETs.

Opacus[2] is a library introduced by Meta AI to integrate DP into PyTorch machine learning pipelines with a vectorized implementation of DP-SGD for high performance. The necessary adjustments can be accomplished with a few lines of code and Opacus makes use of RDP accounting to keep track of the privacy budget spent so far. With

1 `https://github.com/juliecious/CTGAN/blob/683a4d315ce5e05d7152e9d62ecc1a7d08f7df90/fl/main.py#L50` (visited on 02/22/2023)

2 `https://github.com/pytorch/opacus` (visited on 02/23/2023)

TensorFlow Privacy[3] a similar library is also available for TensorFlow that is developed by Google. PyVacy[4] is another alternative for PyTorch, but its runtime cannot compete in benchmarks, as each sample is processed separately in a for-loop to compute the per-sample gradient [You+21].

Furthermore, several open-source implementations of different synthetic data generators are available. *The Synthetic Data Vault Project* (SDV) provides CTGAN[5] as a PyTorch model. SmartNoise Synthesizers,[6] which is maintained by Microsoft in collaboration with Harvard University's OpenDP Initiative, includes PyTorch implementations of DP-CTGAN, PATE-CTGAN, and PATE-GAN.

Liu et al. [Liu+22] recently published a detailed benchmark for FL frameworks called UniFed,[7] which compares nine frameworks in 15 different scenarios. The authors conclude that each framework has advantages and disadvantages in terms of supported models, model performance, training time, communication efficiency, memory consumption, and integration of PETs. A "Framework Selector" is provided on the project website to assist in selecting the appropriate framework for a particular use case. TensorFlow Federated[8] is one of the evaluated frameworks, but at the time of writing does not support deployment to multiple machines. The framework FATE,[9] which is developed by WeBank, is intended for industrial deployment in the cross-silo setting, for instance within a Kubernetes cluster. However, in this work we intend to use a framework that does not require a complex deployment and is also suited for edge devices with low computing power. Flower,[10] on the other hand, can be integrated into arbitrary machine learning frameworks such as PyTorch, TensorFlow or scikit-learn. In addition to the simulation of a FL system on a single machine, deployment to real nodes is supported. Heterogeneous clients do not pose an obstacle, since this framework is designed to be interoperable with different hardware and associated compute, memory, and network resources, as well as programming languages. For example, there is an *software development kit* (SDK) for iOS and Flower has also been successfully utilized with NVIDIA Jetson or Raspberry Pi. Moreover, Flower enables efficient communication between server and clients by using bi-directional *gRPC Remote Procedure Calls* (gRPC) streams and thus binary serialization. In experiments, it was shown that Flower scales to

3 `https://github.com/tensorflow/privacy` (visited on 02/23/2023)
4 `https://github.com/ChrisWaites/pyvacy` (visited on 02/23/2023)
5 `https://github.com/sdv-dev/CTGAN` (visited on 02/24/2023)
6 `https://github.com/opendp/smartnoise-sdk/tree/main/synth` (visited on 02/24/2023)
7 `https://unifedbenchmark.github.io` (visited on 02/24/2023)
8 `https://github.com/tensorflow/federated` (visited on 02/24/2023)
9 `https://github.com/FederatedAI/FATE` (visited on 02/24/2023)
10 `https://github.com/adap/flower` (visited on 02/24/2023)

15 million clients, 1000 of which were selected in each round for concurrently computing the updates [Beu+20].

4 Proposed Method: DP-Fed-CTGAN

In this chapter, we introduce our novel approach for generating synthetic tabular data using FL under strict DP guarantees, which we call DP-Fed-CTGAN. In contrast to existing techniques, the main design principle of DP-Fed-CTGAN is to minimize the amount of data that clients have to disclose about their local datasets to the server, thus adhering to the fundamental goal of FL and reducing the risk of privacy breaches.

First, in Section 4.1, we elaborate on the scenario of a medical data donation, which constitutes the starting point for the development of our method as well as the later evaluation. Moreover, we cover possible attacks that result from the federated setting and explain the countermeasures that we have taken. The architecture of DP-Fed-CTGAN and the steps needed to synthesize data are presented in Section 4.2. Furthermore, the characteristics of the CTGAN model that were briefly mentioned in Section 2.2.2.5 are discussed in greater detail. Finally, in Section 4.3 we address our implementation of DP-Fed-CTGAN, limiting ourselves to the most important aspects.

4.1 Scenario and Threat Model

We made the design decisions of our approach considering the scenario of donating highly vulnerable medical data to research. In such a setup, the clients of the FL system presumably correspond to healthcare providers, while the server is likely operated by a trusted third party such as the Health Data Lab of the BfArM in Germany. We assume that the local datasets consist of tabular medical data that accrue during the treatment of insured persons. With this in mind, as also will be described below, in DP-Fed-CTGAN we have intentionally focused on minimizing the extent of information that the clients have to share with the server, and not to handle the scenario of highly non-IID distributed data especially well, since this would require the server to have extra information about the local datasets in many steps.

For the underlying FL system, we assume that the server is honest-but-curious and the participants are honest. This means that the server will not deviate from the algorithm introduced below, but, for example, an employee of the organization operating the server could try to misuse the information received from the clients to perform an attack that would compromise the privacy of one of the individuals in the local training datasets. We consider an attack to be successful if the adversary succeeds in re-identifying a data donor or gains accurate statistical properties regarding the training data [BM21]. To limit the amount of information that the attacker can learn about an individual from the exchanged model parameters in the worst case, we employ record-level DP by perturbing the local gradients before aggregation. At the same time, these DP guarantees apply to the generator and its synthesized data with respect to inference and model inversion attacks after training.

Combining three PETs for DP-Fed-CTGAN, namely DP, FL, and SDG, together with further reducing data sharing compared to previous works such as Fed-TGAN and HT-Fed-GAN, aims to gain the trust of the public, without which such a data donation cannot succeed, in addition to simply meeting regulatory requirements. However, it should be noted that the risk of a privacy breach can be further reduced, for example, by using *secure multi-party computation* (MPC) in *trusted execution environments* (TEEs), which we leave open for future work [Kai+21].

4.2 Architecture and Algorithms

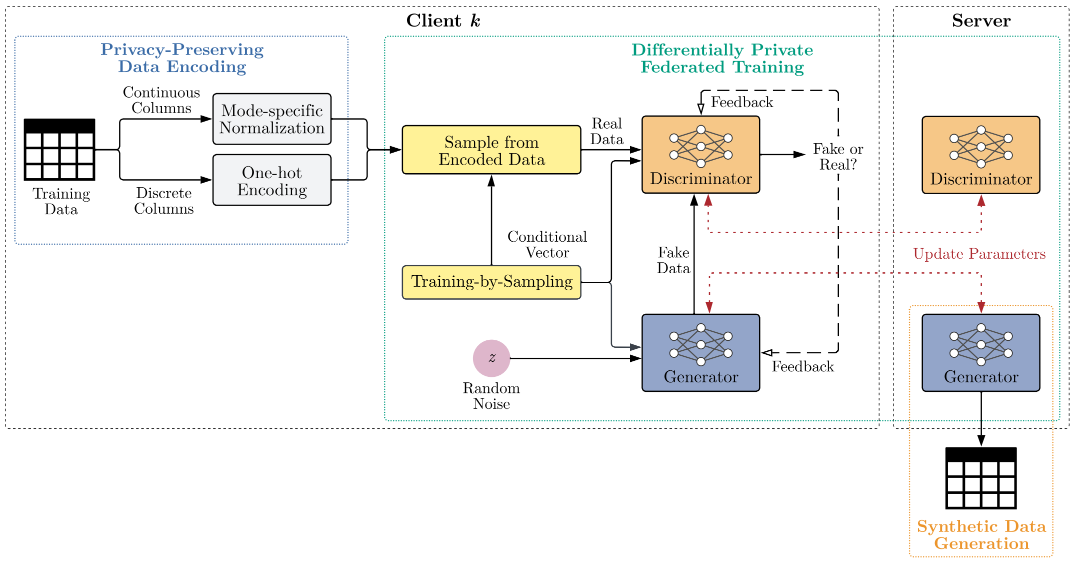


Figure 4.1: Simplified overview of the DP-Fed-CTGAN architecture for one client.

DP-Fed-CTGAN follows the horizontal FL scheme consisting of a central server with the global generator and discriminator and a set of clients that train local generators and discriminators on their respective training dataset. Figure 4.1 illustrates the architecture of our approach and shows the most important steps in the algorithm for generating synthetic data on the server starting from the tabular training data of the client. First, the training data are transformed into a numerical form that facilitates learning the distributions of the column values. During this process, a different encoding is used for continuous and discrete columns. In each federated round, real and fake data are first sampled in order to update the parameters of the discriminator. Afterwards the parameters of the generator are updated. As can be seen in Figure 4.1, the CTGAN framework relies on the training-by-sampling mechanism, i.e., the generator and discriminator additionally receive a conditional vector as input to avoid mode collapse. At the end of each round, every client sends the parameters of the generator and discriminator to the server, where they are aggregated and then returned to the clients.

In the following, we elaborate on the privacy-preserving data encoding process in Section 4.2.1, while Section 4.2.2 covers the actual federated training under DP. Section 4.2.3 addresses the generation of synthetic data in more detail.

4.2.1 Privacy-Preserving Data Encoding

Before the two neural networks in the form of the generator and the discriminator can be trained, the training data must be represented in a suitable way. In the following, we assume that the local training datasets $\{P_1, \dots, P_K\}$ of the K participants consist of the same N_c continuous columns $\{\mathcal{C}_1, \dots, \mathcal{C}_{N_c}\}$ and N_d discrete columns $\{\mathcal{D}_1, \dots, \mathcal{D}_{N_d}\}$. We can thus express a row of a dataset as $r_j = \{c_{1,j}, \dots, c_{N_c,j}, d_{1,j}, \dots, d_{N_d,j}\}$. While discrete columns can be transformed directly into one-hot encoded vectors $\{\mathbf{d}_{1,j}, \dots, \mathbf{d}_{N_d,j}\}$ whose number of components is equal to the number of different categories of the respective column, in CTGAN the so-called mode-specific normalization is performed for continuous columns in order to cope with multimodal distributions of the values, which are often not properly modeled by vanilla GANs. Furthermore, a classical min-max normalization is generally unsuitable for continuous tabular attributes, since long-tailed distributions with extreme outliers are often present, so that the majority of the values are mapped to a very small range at the boundary of the interval $[-1, 1]$. This can cause the gradients to vanish in this range, especially when using the Tanh activation function. We now explain mode-specific normalization first for the centralized scenario [Bis06, pp. 474–486; Xu20]:

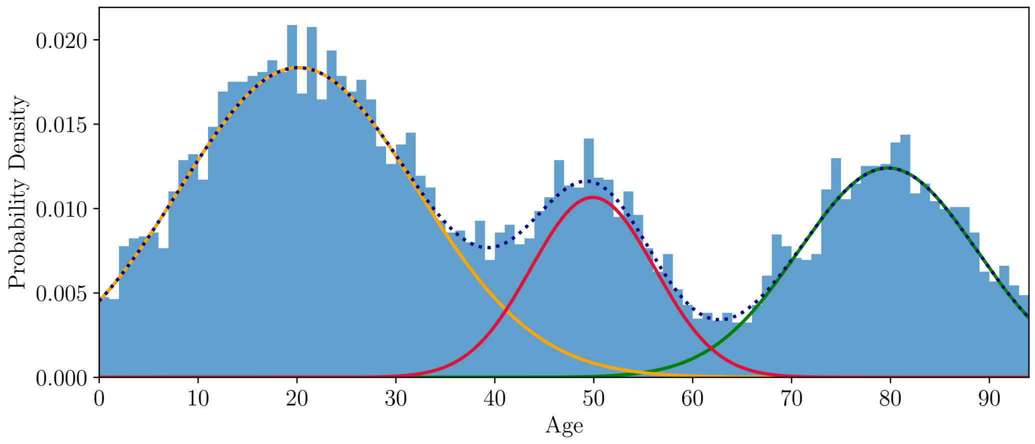


Figure 4.2: Example of a VGM model fitted to a continuous age distribution as part of mode-specific normalization.

1. Fit a *variational Gaussian mixture* (VGM) model to each continuous column $\mathcal{C}_i$ to estimate the number of modes M of the data. Let μ_k denote the mean, Σ_k the variance of the k-th Gaussian mixture, and w_k the weight of the k-th component, then we obtain the *probability density function* (PDF) $p(c_{i,j}) = \sum_{k=1}^{M} w_k \mathcal{N}(c_{i,j} \mid \mu_k, \Sigma_k)$ for the column values. In Figure 4.2, for example, the VGM model finds three modes and the dashed PDF.

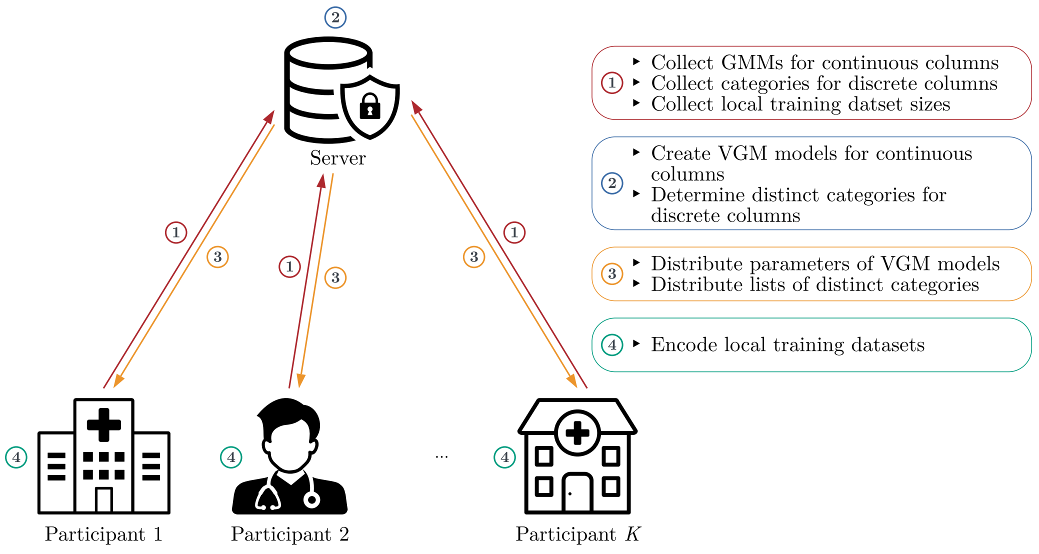
2. For each value $c_{i,j}$ of $\mathcal{C}_i$, calculate the probabilities $\rho_k = w_k \mathcal{N}(c_{i,j} \mid \mu_k, \Sigma_k)$ for $k \in \{1, \dots, M\}$ that $c_{i,j}$ comes from the mode associated with the k-th mixture component.

3. For each $c_{i,j}$, sample a mixture component using the previously calculated probabilities $\rho_1, \dots, \rho_M$. We can now represent $c_{i,j}$ by a one-hot encoded vector $\beta_{i,j}$ indicating the selected mixture component l and thus the mode, and a scalar $\alpha_{i,j} = \frac{c_{i,j} - \mu_l}{4\sqrt{\Sigma_l}}$ specifying the value within the mode by normalization with the mean and standard deviation of the component. For instance, in Figure 4.2 the second mode is most likely chosen for age of 50, so $\beta_{i,j}$ would be $[0, 1, 0]$ and $\alpha_{i,j} = \frac{c_{i,j} - \mu_2}{4\sqrt{\Sigma_2}}$.

With this, we can represent a row by concatenating the continuous and discrete columns:

$$\mathbf{r}_j = \alpha_{1,j} \oplus \beta_{1,j} \oplus \dots \oplus \alpha_{N_c,j} \oplus \beta_{N_c,j} \oplus \mathbf{d}_{1,j} \oplus \dots \oplus \mathbf{d}_{N_d,j}.$$

In the case of a FL architecture, the same value could be represented differently on different clients in the context of mode-specific normalization, since the distribution of



2
Server
1
3
1
3
1
3
1 ‣ Collect GMMs for continuous columns
‣ Collect categories for discrete columns
‣ Collect local training datset sizes
2 ‣ Create VGM models for continuous columns
‣ Determine distinct categories for discrete columns
3 ‣ Distribute parameters of VGM models
‣ Distribute lists of distinct categories
4 ‣ Encode local training datasets
4
Participant 1
4
Participant 2
...
4
Participant K

Figure 4.3: Data encoding process of DP-Fed-CTGAN.

the values of continuous attributes usually varies between clients. For discrete columns, not all categories possibly occur on every client, so the length of the one-hot vectors could be inconsistent. However, the clients must agree on a uniform encoding scheme so that the aggregation of the generator and discriminator parameters can be performed later. Figure 4.3 shows our proposed solution for a uniform encoding of the local training datasets, which at the same time minimizes the sharing of information with the server to protect the privacy of the individuals behind the data.

In the first step, the clients send a list of all distinct values to the server for discrete columns as well as the local training dataset size. The procedure for continuous columns is described in more detail in Algorithm 4.1. Every client fits a *Gaussian mixture model* (GMM) to each continuous column (highlighted in blue), which can optionally be refined in multiple rounds by using the average mixture parameters from Lines 16–20 as initialization for the next round. For this purpose, we use GMMs instead of VGM models, since they have a fixed number of components and are less computationally expensive, while the inferred number of components for VGM models can vary between the clients. Thus, the aggregation of the parameters on the server is simpler. Subsequently, in Lines 22–28, an amount of random samples equal to the size of the local training dataset are drawn from the GMMs of the clients, to which a VGM model is fitted for mode-specific normalization.

In the second step of the data encoding process, the server also determines all distinct

Algorithm 4.1: Federated VGM models.

Input: Number of participants K, maximum number of federated rounds T, local training dataset P_k of each participant, set of continuous columns $\{\mathcal{C}_1, \dots, \mathcal{C}_{N_c}\}$, number of mixture components n

Output: VGM model for each continuous column

```
1  $models \leftarrow \emptyset$;
2  foreach column $c \in \{\mathcal{C}_1, \dots, \mathcal{C}_{N_c}\}$ do
3      foreach round $t \in \{1, \dots, T\}$ do
4          foreach participant $k \in \{1, \dots, K\}$ in parallel do
5              if $t = 1$ then
6                  initialize GMM$_k$ using $k$-means;
7              else
8                  initialize GMM$_k$ with $\mu_1, \dots, \mu_n, \Sigma_1, \dots, \Sigma_n, w_1, \dots, w_n$;
9              end
10             fit GMM$_k$ to $P_k\,[c]$;      ▷ Fit GMM to column c of local dataset
11             $\mu_1^k, \dots, \mu_n^k \leftarrow$ (means of GMM$_k$ in ascending order);
12             $\Sigma_1^k, \dots, \Sigma_n^k \leftarrow$ (variances of GMM$_k$ sorted according to means);
13             $w_1^k, \dots, w_n^k \leftarrow$ (weights of GMM$_k$ sorted according to means);
14         end
15         if $t \neq T$ then
16             for $1 \leq j \leq n$ do
17                 $\mu_j \leftarrow \frac{1}{K} \sum_{k=1}^{K} \mu_j^k$;
18                 $\Sigma_j \leftarrow \frac{1}{K} \sum_{k=1}^{K} \Sigma_j^k$;
19                 $w_j \leftarrow \frac{1}{K} \sum_{k=1}^{K} w_j^k$;
20             end
21         else
22             $list \leftarrow \emptyset$;
23             foreach participant $k \in \{1, \dots, K\}$ do
24                 $samples \leftarrow$ (take $|P_k|$ random samples from GMM with parameters
                     $\mu_1^k, \dots, \mu_n^k, \Sigma_1^k,, \dots, \Sigma_n^k, w_1^k,, \dots, w_n^k$);
25                 extend $list$ with $samples$;
26             end
27             $model_c \leftarrow$ (fit VGM model to $list$);
28             append $model_c$ to $models$;
29         end
30     end
31 end
32 return $models$;
```

categories for discrete columns. Next, the parameters of the learned VGM models of each continuous column as well as the list of all distinct categories for discrete columns are sent to the clients. Eventually, the local training datasets are transformed into a suitable representation for training using mode-specific normalization and one-hot encoding as described above. It should be noted that this transformation can be easily reversed.

4.2.2 Differentially Private Federated CTGAN Training

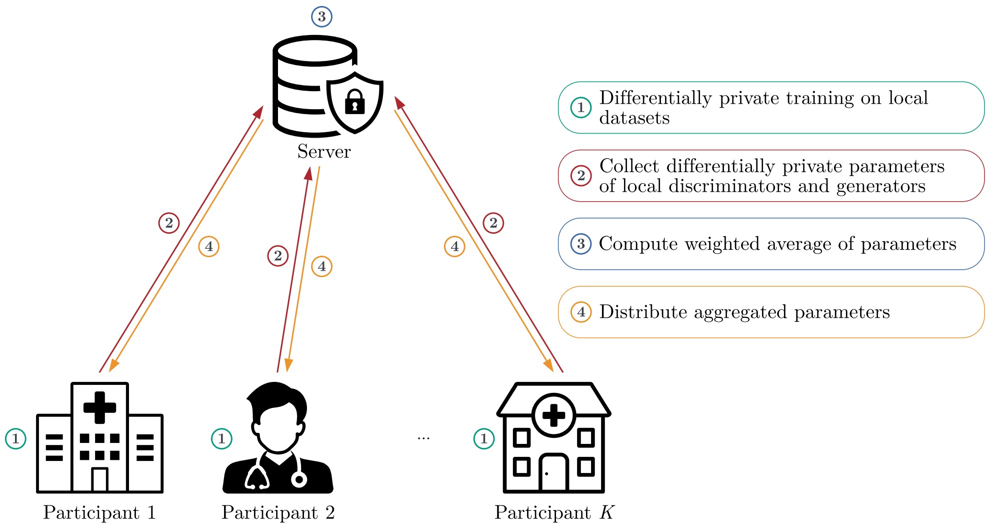

3
Server
2
4
2
4
4
2
1
Participant 1
1
Participant 2
...
1
Participant K
1 Differentially private training on local datasets
2 Collect differentially private parameters of local discriminators and generators
3 Compute weighted average of parameters
4 Distribute aggregated parameters

Figure 4.4: Training process of DP-Fed-CTGAN.

As shown in Figure 4.4, the procedure of a federated training round can be divided into four steps. First, each client trains the discriminator and generator on the local training dataset in a differentially private manner by privatizing the gradients of the discriminator as in DP-SGD. Afterwards, the clients send the local parameters to the server, where they are aggregated by computing the weighted average. Finally, the aggregated parameters are sent back to the clients, forming the starting point for the parameter update in the next round.

We now discuss the client-side training process of DP-Fed-CTGAN in depth, as depicted in Algorithm 4.2. First, in Line 1, the number of steps per local epoch is calculated by dividing the local training dataset size by the batch size. At the beginning of each epoch (Lines 2–5), we determine whether the privacy budget is already exhausted by querying the RDP-based privacy accountant $\mathcal{A}$ and comparing the output to ε_{target}.

If the budget does not allow any further training, the current parameters of the generator and discriminator are returned. Otherwise, we proceed with the training by creating a batch of so-called conditional vectors with the training-by-sampling mechanism in Line 8, since CTGAN builds on the conditional GAN (cGAN) architecture.

Algorithm 4.2: Client-side training algorithm of DP-Fed-CTGAN.

```
Input: Current global parameters $\theta_g$ and $\theta_d$ of generator $G$ and discriminator $D$,
       local training dataset $P_k$ of $k$-th participant, number of discrete columns
       $N_d$ in $P_k$, number of local epochs $E$, learning rate $\eta$, mini-batch size $m$,
       gradient norm bound $C$, noise scale $\sigma$, privacy budget $(\varepsilon_{target}, \delta_{target})$
Output: Updated parameters $\theta_g$ and $\theta_d$ of differentially private generator and
        discriminator
1  $S \leftarrow \max\left(1, \frac{|P_k|}{m}\right)$;                ▷ Number of steps per local epoch
2  foreach local epoch $e \in \{1, \ldots, E\}$ do
3      $\varepsilon \leftarrow$ (query RDP accountant $\mathcal{A}$ with $\delta_{target}$);
4      if $\varepsilon > \varepsilon_{target}$ then
5          return $\theta_g, \theta_d$;                         ▷ Privacy budget is consumed
6      foreach step $s \in \{1, \ldots, S\}$ do
7          for $1 \leq j \leq m$ do
8              create conditional vector $cond_j$ with training-by-sampling;
9              $z_j \sim \mathcal{N}(0, I)$;                    ▷ Sample from noise prior
10             $\hat{\mathbf{r}}_j \leftarrow G(z_j, cond_j)$;   ▷ Generate fake data
11             $\mathbf{r}_j \sim \text{Uniform}(P_k \mid cond_j)$;  ▷ Sample real data
12         end
13         $\mathcal{L}_d \leftarrow \frac{1}{m}\sum_{i=1}^{m} \text{BCE}(D(\hat{\mathbf{r}}_i, cond_i), 0) + \frac{1}{m}\sum_{i=1}^{m} \text{BCE}(D(\mathbf{r}_i, cond_i), 1)$;
14         $\theta_d \leftarrow \theta_d - \eta \cdot \text{DPAdam}(\nabla_{\theta_d}\mathcal{L}_d, C, \sigma)$;
15         for $1 \leq j \leq m$ do
16             create conditional vector $cond_j$ with training-by-sampling;
17             $z_j \sim \mathcal{N}(0, I)$;                    ▷ Sample from noise prior
18             $\hat{\mathbf{r}}_j \leftarrow G(z_j, cond_j)$;   ▷ Generate fake data
19         end
20         $\mathcal{L}_g \leftarrow \frac{1}{m}\sum_{i=1}^{m} \text{BCE}(D(\hat{\mathbf{r}}_i, cond_i), 1)$;
21         $\theta_g \leftarrow \theta_g - \eta \cdot \text{Adam}(\nabla_{\theta_g}\mathcal{L}_g)$;
22     end
23 end
24 return $\theta_g, \theta_d$;
```

These conditional vectors, which indicate the category of a particular discrete column, are additionally given as input to the generator and discriminator to avoid mode collapse for imbalanced categorical columns. In the following, we briefly explain the steps of training-by-sampling as presented in the CTGAN framework [Xu+19]:

1. For each discrete column $\mathcal{D}_i$ with $i \in \{1, \ldots, N_d\}$, create a zero-filled mask vector $\mathbf{m}_i = \left[m_i^{(1)} \ldots m_i^{(|\mathcal{D}_i|)}\right]$ so that each component corresponds to a category in the list of the $|\mathcal{D}_i|$ possible categories received from the server.

2. Randomly select one of the discrete columns with equal probability, which we refer to as $\mathcal{D}_{i^*}$.

3. Calculate a PMF for the selected column $\mathcal{D}_{i^*}$ such that the probability mass of each category is equal to the logarithm of its frequency.

4. Randomly choose a category k^* based on the PMF just calculated and set the k^*-th component of the i^*-th mask to one, i.e., $\mathbf{m}_{i^*}^{(k^*)} = 1$.

5. Compute the conditional vector $cond = \mathbf{m}_1 \oplus \ldots \oplus \mathbf{m}_{i^*} \oplus \ldots \oplus \mathbf{m}_{N_d}$. As an example, suppose we have a table with two discrete columns containing the values $\mathcal{D}_1 = \{A, B\}$ and $\mathcal{D}_2 = \{C, D, E\}$, respectively. Assuming $i^* = 2$ and $k^* = 1$, we would obtain the masks $\mathbf{m}_1 = [0, 0]$ and $\mathbf{m}_2 = [1, 0, 0]$, thus resulting in $cond = [0, 0, 1, 0, 0]$.

Compared to HT-Fed-GAN, we do not consider the global frequencies of the categories, but rely on the local frequencies in the respective training dataset of the client. To compute the global frequencies, the clients would have to communicate the local frequencies to the server, which we believe would pose an unnecessary threat to the privacy of the individuals in the underlying attacker model and would contradict the goal of FL to minimize the amount of information that clients have to share about the local datasets.

Continuing with the for loop of Algorithm 4.2 (Lines 9–11), we draw random noise from a multivariate normal distribution, which the generator receives along with the conditional vector to synthesize fake data. Furthermore, we uniformly draw a sample among the rows from the local dataset that satisfy the constraints of the corresponding conditional vector $cond_j$. In Lines 13–14, we then compute the loss of the discriminator as the average BCE loss between the label predicted by the discriminator and the actual label for the batches of fake as well as real data. Unlike the original training algorithm of CTGAN, no additional term enforcing the 1-Lipschitz condition of WGAN that penalizes the norm of the gradient is added to the loss, since the per-sample gradients have to be clipped anyway for the DP guarantees. To update the discriminator parameters, we use a differentially private version of the Adam optimizer, i.e., the Adam algorithm introduced by Kingma and Ba [KB14] is adjusted to include per-sample gradient clipping and the injection of noise to the average gradient.

Then another batch of fake data is generated and assessed by the discriminator, leading to the generator's loss (Lines 15–20). Updating the parameters of the generator with Adam thereafter is also differentially private according to Theorem 2.6 without adding noise (Line 21), since the output of the discriminator is differentially private and the generator has no access to the real data. Moreover, to comply with DP, a penalty term from the original CTGAN framework to prevent the violation of the conditional vector is not included in the loss function of the generator. Lastly, the parameters of the generator and discriminator are returned after the specified number of local epochs.

Algorithm 4.3: Server-side training algorithm of DP-Fed-CTGAN.

```
Input: Number of participants $K$, maximum number of federated rounds $T$,
       local training dataset size $|P_k|$ of each participant
Output: Parameters $\theta_g$ of differentially private generator
1  initialize $\theta_g$ and $\theta_d$;
2  foreach participant $k \in \{1, \dots, K\}$ do
3      $w_k = \frac{|P_k|}{\sum_{j=1}^{K} |P_j|}$
4  end
5  foreach round $t \in \{1, \dots, T\}$ do
6      foreach participant $k \in \{1, \dots, K\}$ in parallel do
7          $\theta_g^k, \theta_d^k \leftarrow$ (execute Algorithm 4.2 on participant $k$ with current global
           parameters $\theta_g$ and $\theta_d$);
8      end
9      $\theta_g \leftarrow \sum_{k=1}^{K} w_k \theta_g^k$;
10     $\theta_d \leftarrow \sum_{k=1}^{K} w_k \theta_d^k$;
11 end
12 return $\theta_g$;
```

On the other hand, as can be seen in Algorithm 4.3, the parameters of the generator and discriminator are initialized on the server at the beginning. The weights of the clients, which are later used to aggregate the parameters, are given by the local dataset size in relation to the total number of samples (Lines 2–4). We decided against a weighting scheme as in Fed-TGAN, which tries to compensate for a non-IID distribution of the data across clients, since it requires the knowledge of the category frequencies by the server. We will examine the cost of this approach in terms of utility and fidelity in the evaluation. Next, starting with the current global parameters of the generator and discriminator, the updated parameters are retrieved from all clients in each round and aggregated according to the weights previously calculated. When the desired number of federated rounds are completed, the parameters of the generator are returned.

4.2.3 Generation of Synthetic Data

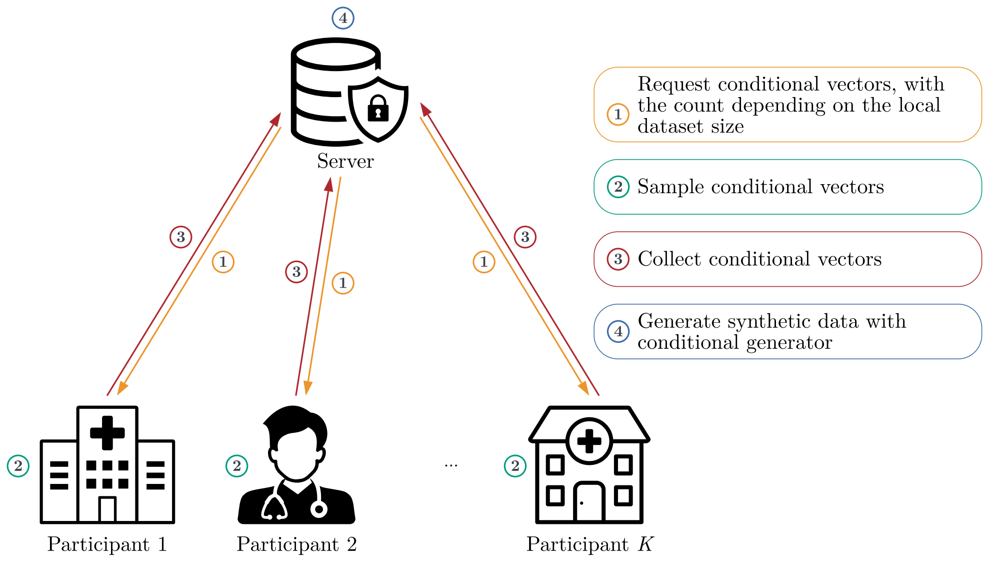

4
Server
Request conditional vectors, with the count depending on the local dataset size
Sample conditional vectors
Collect conditional vectors
Generate synthetic data with conditional generator
Participant 1
Participant 2
...
Participant K

Figure 4.5: Data generation process of DP-Fed-CTGAN.

Now that the training process is finished, the generation of synthetic data can be performed as shown in Figure 4.5. Therefore, let us assume that we want to synthesize n rows. In the first step, the server requests $n \cdot w_k$ conditional vectors from client k, where the w_k are the weights calculated in Algorithm 4.3. This differentiates our algorithm from previous works like Fed-TGAN and HT-Fed-GAN, because in these approaches the server knows the frequencies of all categories and therefore the conditional vectors can be constructed on the server. Our proposed solution is thus more privacy-friendly, but faces higher communication costs.

After that, the clients create the desired number of conditional vectors using the training-by-sampling method we explained earlier, but with the difference that the PMF uses the actual frequencies of the values rather than their logarithm. Once the server received the conditional vectors in the third step, synthetic rows can be generated by feeding the generator with noise from a multivariate normal distribution and the conditional vectors. Lastly, the encoding of the data is reversed.

4.3 Implementation

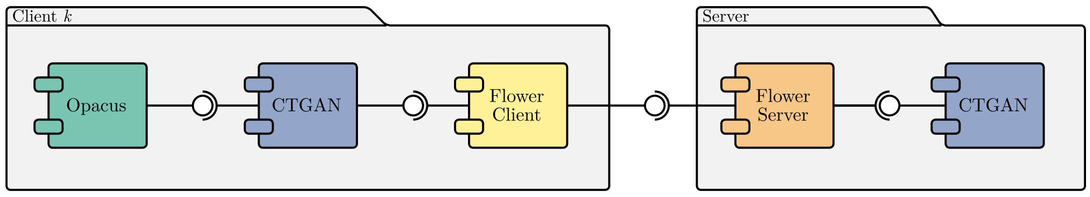


Figure 4.6: Software components of DP-Fed-CTGAN.

Our implementation of DP-Fed-CTGAN is mainly based on the open-source frameworks CTGAN from SDV, Opacus, and Flower, which were already outlined in Section 3.2. Figure 4.6 illustrates the relationships between these software entities as a UML component diagram. We begin with our customized CTGAN library, which provides the functionality for the local training on the clients as well as generating synthetic data on the server. First, we made the necessary adjustments to the original CTGAN implementation for our privacy-preserving data encoding and the sampling of conditional vectors in the federated setting. To ensure record-level DP, we integrated Opacus by attaching a `PrivacyEngine` to the Adam optimizer of the discriminator. Regarding the other required changes in the `train` method to achieve compatibility with Opacus, we followed the DP-CTGAN implementation from SmartNoise. Since in this process the loss functions were modified to match Algorithm 4.2, a Sigmoid activation is added as the last layer of the discriminator, as shown in Figure 4.7.

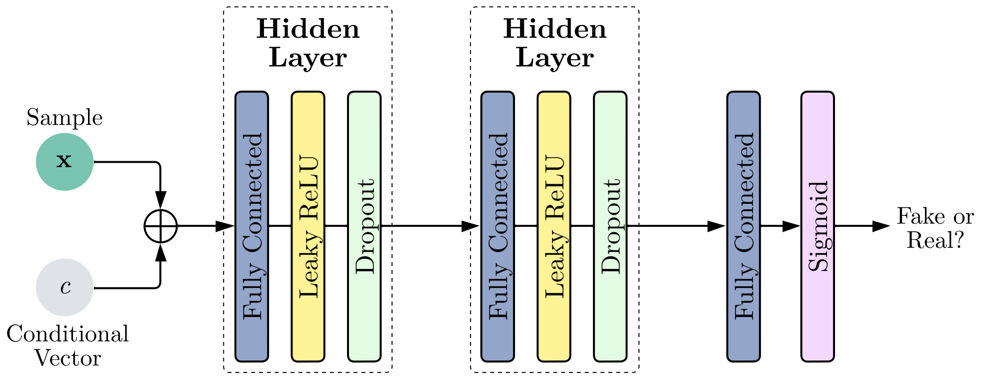


Figure 4.7: Structure of the discriminator in DP-Fed-CTGAN.

For completeness, the structure of the generator network is visualized in Figure 4.8, where we would like to highlight the concatenative skip connections and the use of different activation functions in the output layer depending on the vector component. Namely, for the scalar values α the Tanh activation is used, while for the one-hot vectors β and $\mathbf{d}$ the tailor-made Gumbel softmax is used [JGP16].

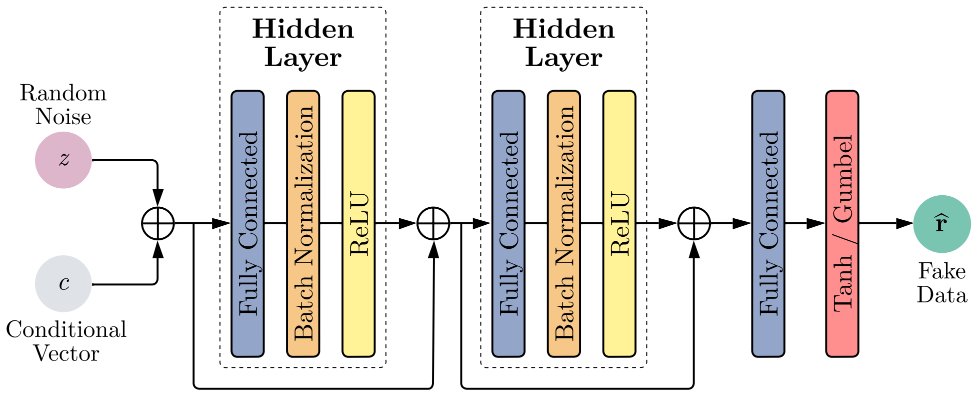


Figure 4.8: Structure of the generator in DP-Fed-CTGAN.

Building a FL system with Flower is as simple as implementing the `NumPyClient` interface for the Flower Clients and specifying a `Strategy` object for the Flower Server that allows customization of the learning process. Once the Flower Server and the Flower Clients are started, the library handles the communication between them and invokes the appropriate functions at the right time. The functionality of a `NumPyClient` is implemented in the three methods `get_parameters`, `set_parameters`, and `fit`. In our case, the `fit` method receives the aggregated parameters from the server and executes the `set_parameters` function to update the local generator and discriminator, after which the local training is performed by calling the `train` method of CTGAN. Ultimately, through `fit` the `get_parameters` function is called and the model updates are sent to the server.

In comparison, there are two important methods in the `Strategy` class for our setting. `configure_fit` selects the clients that will participate in the upcoming round, which in our case are all connected clients. Before the first round, this method also retrieves the information needed for the data encoding (see Section 4.2.1) and transmits the final encoding scheme to the clients. To this end, we use the so-called Flower Criteria, because at the time of writing there is no other way for retrieving information from the clients prior to the actual training. On the other hand, the aggregation of the parameters that the server receives by calling `fit` on each client is implemented in `aggregate_fit`. The aggregated parameters of the local generators are also assigned to the server-side generator. After the federated training is completed, this generator allows to synthesize data as described in Section 4.2.3, whereby the required conditional vectors are also retrieved from the clients via a Flower Criterion.

5 Evaluation

In this chapter, we perform a comprehensive evaluation of DP-Fed-CTGAN. After detailing the experimental setup in Section 5.1, we provide the results of our experiments in Section 5.2. Ultimately, in Section 5.3, we will summarize the key findings, but also address the limitations of our approach and suggest possible directions for future research. All source code arising from the implementation of our method as well as all Juypter notebooks from the evaluation are available in a GitLab repository.[1]

5.1 Experimental Setup

The following section first introduces the metrics we selected to benchmark DP-Fed-CTGAN as well as the datasets that served as the basis for our experiments. Next, we outline the baselines that enable a comparison of the performance of our method and discuss the hyperparameters we used. Last, we present the execution environment for the experiments.

5.1.1 Metrics

In the literature, the quality of the generated synthetic data is usually assessed in the dimensions of utility and fidelity. Utility refers to the performance of using synthetic data in place of real data for a given downstream task, with the primary use case being classification algorithms. In contrast, for fidelity, we examine whether the synthesizers preserve the statistical properties of the data.

5.1.1.1 Utility

To assess the utility of synthetic data, the general idea is to compare the two techniques, *train on real, test on real* (TRTR) and *train on synthetic, test on real* (TSTR), by means of common classification metrics. For this, we use the framework shown in Figure 5.1, which was originally proposed in [EHR17]. Here, the real dataset is

1 `https://gitlab.com/leitmori/DP-Fed-CTGAN` (visited on 06/15/2023)

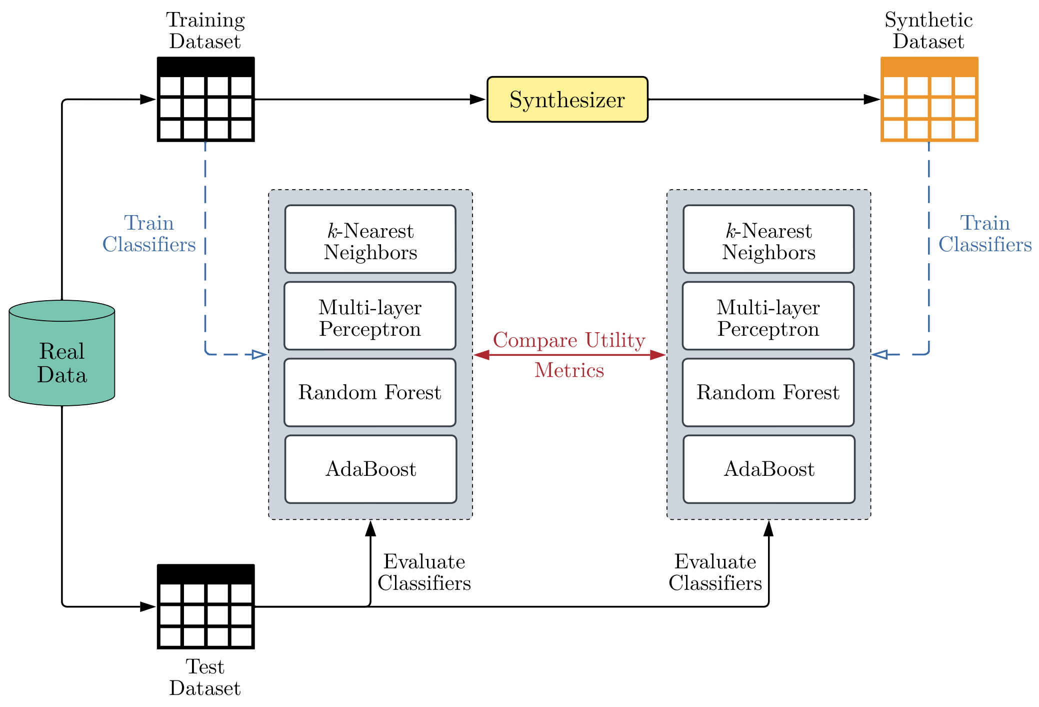


Figure 5.1: Framework for the utility evaluation.

first divided into a training dataset and a test dataset, for which the ratio 80/20 is usually employed. Subsequently, various classifiers are trained on the training dataset. We used the following algorithms from the scikit-learn library, leaving the default settings for the most part: (i) `KNeighborsClassifier` with `n_neighbors=10`, (ii) `MLPClassifier` with `hidden_layer_sizes=(100,)`, (iii) `RandomForestClassifier` with `n_estimators=100`, and (iv) `AdaBoostClassifier` with `n_estimators=50`. Furthermore, the synthesizer to be evaluated is also trained with the training dataset and a synthetic dataset with the same number of records is generated. Subsequently, the same classifiers are trained on the synthetic dataset. Finally, the classifiers can be evaluated on the test data and their performance can be compared using classification metrics.

The starting point for the classification metrics is the so-called confusion matrix as shown in Figure 5.2, which allows a structured representation of the decisions of a classifier. For this purpose, a distinction is made between four options. A *true positive* (TP) is a correctly as positive classified sample, while a *true negative* (TN) is a sample correctly predicted as negative. Moreover, a sample incorrectly labeled as positive is a *false positive* (FP) and a sample erroneously classified as negative is a *false negative*

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Figure 5.2: Confusion matrix.

(FN). With these quantities, we can define the accuracy as follows [LKA16]:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

However, an evaluation of a classifier based on accuracy alone can be misleading in the case of class imbalance. In addition, the accuracy score reveals little about the ability of a model to discriminate between positive and negative samples. We therefore consider two more metrics with *area under the receiver operating characteristic curve* (AUROC) and *area under the precision-recall curve* (AUPRC), for which we must first define the quantities *true positive rate* (TPR), *false positive rate* (FPR), and precision that can be derived from the confusion matrix [Tha20]:

$$\text{TPR / Recall} = \frac{TP}{TP + FN}, \qquad \text{FPR} = \frac{FP}{FP + TN}, \qquad \text{Precision} = \frac{TP}{TP + FP}.$$

To construct the ROC curve, for different decision thresholds the corresponding FPR is plotted along the x-axis and the TPR is plotted along the y-axis. For the AUROC metric, which corresponds to the probability that a randomly chosen positive sample is assigned a higher score by the model than a randomly chosen negative sample, the area under the ROC curve is calculated [Bra97; LKA16]. In the case of highly imbalanced classes, AUPRC is an insightful metric as it corresponds to the area under the precision-recall curve, which illustrates the trade-off between precision and recall for varying decision thresholds [Tha20].

5.1.1.2 Fidelity

To examine the fidelity, usually the differences between the distributions of the attribute values and between the relationships of the attributes are considered. We compute the distance from the probability distributions of the discrete attributes of the training dataset to the distributions of the corresponding attributes in the synthetic dataset using the *Jensen-Shannon distance* (JSD) as follows [Her+23]:

$$\mathrm{JSD}(P, Q) = \sqrt{\frac{D_{KL}(P \parallel M) + D_{KL}(Q \parallel M)}{2}},$$

where $M = \frac{1}{2}(P + Q)$ and D_{KL} is the *Kullback-Leibler divergence* (KLD). Next, we calculate the average of the distances of all categorical columns and obtain a metric between zero and one, where a lower value corresponds to a higher similarity of the distributions. For continuous attributes, we resort to the *Wasserstein-1 distance* (WD), which is also known as *earth mover's distance* (EMD), since the KLD can be ill-defined if the probability distributions of the real and synthetic attribute values do not overlap [ACB17; Zha+21]:

$$\mathrm{WD}(P, Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\|\right],$$

where $\Pi(P, Q)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are P and Q. Illustratively, the WD indicates the minimum cost to transform the distribution P into the distribution Q by transporting probability mass. To enable a meaningful aggregation of the distances of all continuous columns into one score, we first perform a min-max normalization for each attribute of the training dataset and apply the same transformation to the synthetic dataset.

We further assess the ability of the synthesizers to learn the relations between each pair of attributes. On the one hand, we use Cramér's V, which is a measure of association between categorical variables based on the χ^2 (chi-squared) test. On the other hand, we analyze the dependencies between continuous attributes with the Pearson correlation coefficient [Her+23].

5.1.2 Datasets

For our experiments described below, we selected three datasets commonly used for generating synthetic data from the *University of California, Irvine* (UCI) Machine Learning Repository and three real-world medical datasets from Kaggle. All datasets contain both continuous and discrete attributes. Two datasets, Covertype and Obesity, are multiclass classification problems. Table 5.1 gives an overview of the properties of the datasets, which we now present in more detail:

Adult: The 48842 records of the Adult dataset were extracted from the 1994 US Census database. The task is to predict for each individual, based on 14 features such as age, education, and hours of work per week, whether they have an annual income greater than \$50,000 [BK96].

Bank Marketing: This dataset contains information about a marketing campaign of a Portuguese bank. The objective of the classification task is to predict whether the customer has opened a term deposit [MCR14].

Covertype: The Covertype dataset contains U.S. Forest Service (USFS) observations such as elevation, aspect, slope, hillshade, soil type, and one of seven possible forest cover types for 581012 30-by-30 meter cells in the Roosevelt National Forest [Bla98].

Cardiovascular Disease: The Cardiovascular Disease dataset[2] from Kaggle contains 70000 patients with 11 features such as age, weight, and blood pressure in order to predict whether they suffer from a cardiovascular disease.

Diabetes: The Pima Indians Diabetes Database,[3] originally from the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), consists of 764 diagnostic measurements for predicting whether or not a patient has diabetes.

Obesity: The Obesity dataset[4] originates from a paper that addresses the estimation of obesity levels based on eating habits and physical condition, specifically the task of assigning one of seven obesity levels to the 2111 individuals.

Dataset	Number of Rows		Number of Attributes		Target Variable
	Training	Testing	Continuous	Discrete	
Adult	32561	16281	6	9	income
Bank Marketing	36168	9043	6	11	y
Covertype	464809	116203	10	45	Cover_Type
Cardiovascular Disease	56000	14000	5	7	cardio
Diabetes	614	154	7	2	Outcome
Obesity	1688	423	8	9	NObeyesdad

Table 5.1: Properties of the datasets used for the experiments.

5.1.3 Baselines and Hyperparameters

We compare our proposed method, DP-Fed-CTGAN, with three state-of-the-art GAN-based tabular data synthesizers that were already introduced in Chapter 3. Since the

2 `https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset` (visited on 05/19/2023)

3 `https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database` (visited on 05/19/2023)

4 `https://www.kaggle.com/datasets/ankurbajaj9/obesity-levels` (visited on 05/22/2023)

majority of previous approaches in the federated setting do not provide DP guarantees or have only been evaluated with debatable privacy parameters, we also consider a version of our synthetic data generator without DP in the experiments, which we call Fed-CTGAN. In the following, we briefly summarize the selected hyperparameters of the baseline synthesizers with the exception of the batch size and the number of epochs, for which we determined appropriate values for each dataset that can be found in the Jupyter notebooks in the GitLab repository:

CTGAN: For the non-private centralized CTGAN implementation from SDV, we chose the default values for all hyperparameters.

DP-CTGAN: This differentially private version of CTGAN from the SmartNoise Synthesizers project, like our approach, uses Opacus for gradient perturbation and privacy accounting. The number of training epochs is determined by the privacy budget (ε, δ) as well as the noise scale σ. Table 5.2 shows the privacy configuration we picked for each dataset, while we kept the default values for the remaining hyperparameters.

Fed-TGAN: Since the authors did not publish the code of their technique, we re-implemented Fed-TGAN according to the details stated in the paper [Zha+21]. Apart from the missing integration of DP, the main difference to our approach is that the clients have to provide the server with the frequencies of the individual categories of each discrete column. Therefore, the server can create the conditional vectors needed for synthesizing records itself and does not need to retrieve them from the clients. Furthermore, the authors use this information when aggregating the model parameters of the clients, while in our approach the weights for the aggregation depend only on the respective local dataset sizes.

Dataset	ε	δ	σ **of DP-CTGAN**	σ **of DP-Fed-CTGAN**
Adult	3	1e-5	2	2
Bank Marketing	3	1e-5	2	2
Covertype	3	1e-6	1	2
Cardiovascular Disease	3	1e-5	2	2
Diabetes	3	1e-3	2	2
Obesity	3	1e-4	3	3

Table 5.2: Configuration of Opacus for DP-CTGAN and DP-Fed-CTGAN.

Since tuning all the hyperparameters of (DP-)Fed-CTGAN would not have been possible in a reasonable amount of time with the computational resources at our disposal, we only experimented with minor changes starting from default values of CTGAN. The most important hyperparameters that we finally used for the evaluation are listed in Table 5.3. Again, we refer to the Jupyter notebooks for the batch size and the number of FL rounds, while the configuration of Opacus for the different datasets can be seen in Table 5.2.

Step	**Hyperparameter**	**Fed-CTGAN**	**DP-Fed-CTGAN**
Encoding	Mixture components	10	10
	GMM rounds (Algorithm 4.1)	5	5
Training	Learning rate of generator	2e-4	2e-4
	Learning rate of discriminator	2e-4	2e-4
	Weight decay of generator	1e-5	1e-5
	Weight decay of discriminator	1e-5	1e-5
	Betas of Adam	0.5, 0.9	0.5, 0.9
	PAC size [Lin+17]	10	1
	Discriminator updates per generator update	3	1
	Local epochs per federated round	3	3

Table 5.3: Key hyperparameters of Fed-CTGAN and DP-Fed-CTGAN.

5.1.4 Experimental Environment

Given that there were a total of 764 synthetic datasets to be generated for the evaluation, we split the experiments between two testbeds. Our main machine running CentOS 7 and Python 3.10.11 was equipped with two Intel Xeon 4210 2.2 GHz CPUs, eight

Library	**Version**
Flower	1.3.0
PyTorch	1.13.1
Opacus	0.14.0
scikit-learn	1.2.2
pandas	1.5.3
NumPy	1.24.3
SciPy	1.10.1
CTGAN	0.7.3
SmartNoise Synthesizers	0.3.7

Table 5.4: Version numbers of the main libraries.

NVIDIA GeForce RTX 2080 Ti GPUs, and 144 GiB of RAM. In addition, we deployed a Google Cloud *virtual machine* (VM) running Debian 11 and Python 3.10.10 with eight vCPUs (Intel Xeon 2.3 GHz), an NVIDIA Tesla P100 GPU, and 30 GiB of RAM. The exact same versions of the required Python libraries were installed on both testbeds, of which we list the most important ones in Table 5.4.

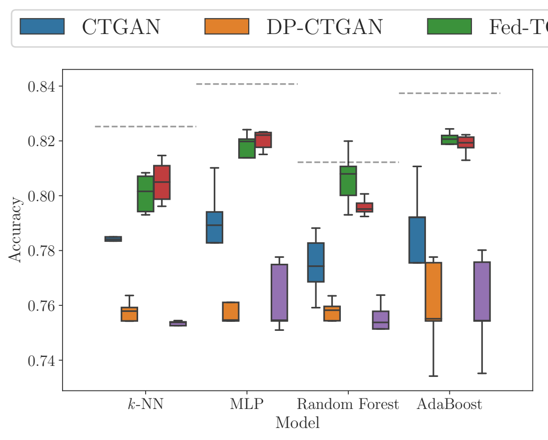
5.2 Results

Below, we first present the evaluation results in terms of the utility of the synthetic data. Next, we analyze how realistic the output of the different synthesizers is by means of the fidelity metrics already introduced. To determine whether our approach of minimizing the amount of information that clients have to share about their local records has a negative impact on performance, we also consider the scenario of non-IID distributed data between the clients. Additionally, we investigate the influence of the privacy budget and the number of clients on utility and fidelity. We conducted five runs for each experiment to achieve more robust results, distributing the data randomly among the clients in the federated setting for each run. Unless otherwise specified, we used three clients and $\varepsilon = 3$ as the privacy budget for the differentially private synthesizers.

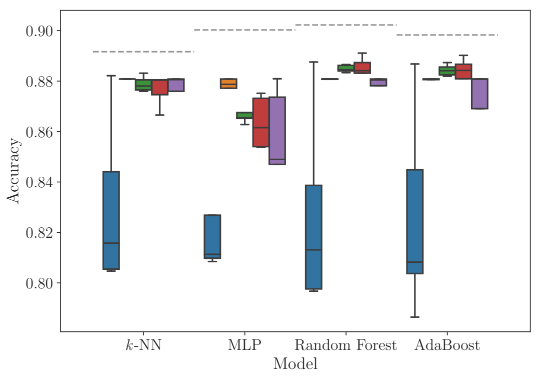
5.2.1 Utility

We begin by examining the utility of the data generated by different synthesizers in terms of the accuracy of the trained classifiers. Figure 5.3 visualizes the results for all datasets, with the four different classifiers *k-nearest neighbors* (k-NN), *multilayer perceptron* (MLP), random forest, and AdaBoost plotted along the x-axis and the accuracy along the y-axis. For comparison, the average accuracy scores of TRTR are shown as a dashed gray line for each classifier. It should be noted that our primary concern is to analyze the differences between TSTR and TRTR, rather than the values themselves.

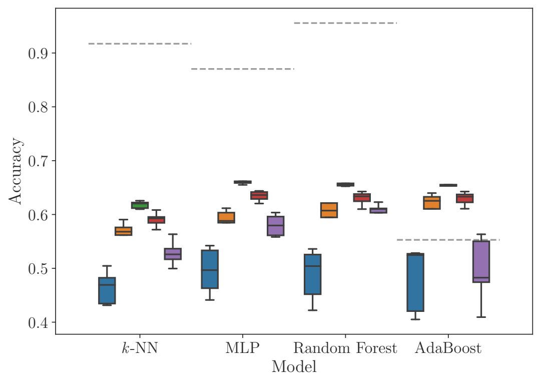
For the Adult dataset in Figure 5.3a, the loss in utility is smallest for the non-private federated methods (Fed-TGAN and Fed-CTGAN) and is essentially negligible. In the case of random forest, the upper whisker of Fed-TGAN is even above the average of the accuracy for the real dataset. As expected from the perturbation of the gradients during training, the two differentially private methods (DP-CTGAN and DP-Fed-CTGAN) have a lower utility and show a similar performance for all classifiers. The median accuracy of CTGAN, and thus the centralized scenario without privacy guarantees, falls between the two groups.



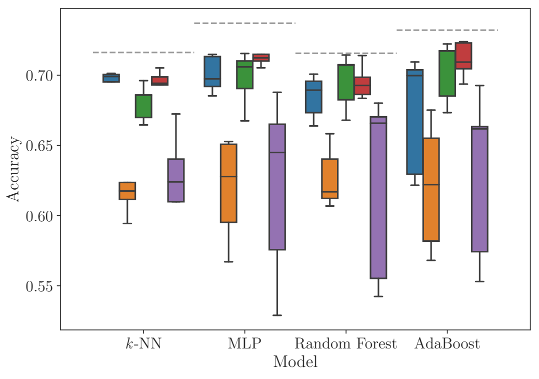
(a) Adult



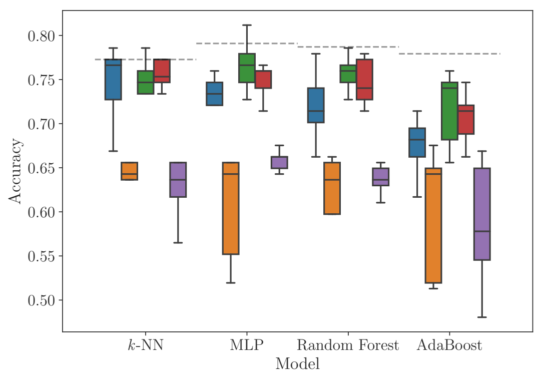
(b) Bank Marketing



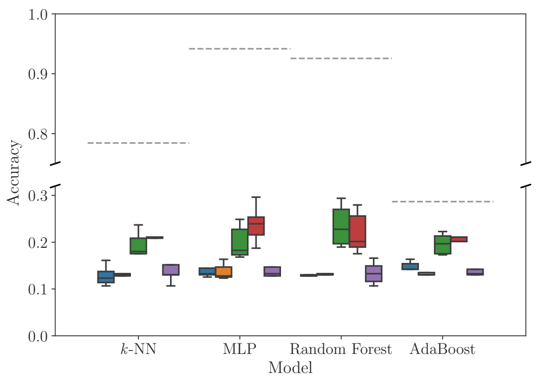
(c) Covertype



(d) Cardiovascular Disease



(e) Diabetes



(f) Obesity

Figure 5.3: Comparison of accuracy scores for different classifiers and SDG methods for each dataset.

Unlike the other datasets, the Bank Marketing dataset has a significant class imbalance, with about 88 percent of the samples belonging to the negative class. For this reason, as seen in Figure 5.3b, there are only minor differences in accuracy between the individual synthesizers, since the classifiers learn mostly the negative classes. However, it is evident that CTGAN consistently achieves the lowest median accuracy and also has the largest variabilities. The results for AUROC and AUPRC discussed later permit a better utility assessment for this dataset.

The multiclass classification problem of the Covertype dataset in Figure 5.3c results in a substantial decline in accuracy in the case of TSTR, except for AdaBoost. Similar to Figure 5.3a, the non-private federated methods achieve the highest accuracy, with Fed-TGAN performing slightly better than Fed-CTGAN. DP-CTGAN follows with only a small offset, while the disparity with DP-Fed-CTGAN and especially CTGAN is higher. Excluding AdaBoost, the median accuracy of CTGAN is clearly below the median of DP-Fed-CTGAN.

A grouping of accuracy values between non-private and differentially private synthesizers can also be observed for the Cardiovascular Disease dataset in Figure 5.3d. The use of synthetic data here comes at almost no cost in terms of utility for CTGAN, Fed-TGAN, and Fed-CTGAN. But even for the differentially private GANs, the loss of accuracy is less than 10 percent in the majority of the runs, although the variability is larger. Furthermore, the median score of our method is continuously above DP-CTGAN.

We observe similar results for the Diabetes dataset in Figure 5.3e. Here, the whiskers of the non-private synthesizers for several classifiers are higher than the corresponding average accuracy when using the real data. DP-CTGAN and DP-Fed-CTGAN score comparably with the exception of AdaBoost, where the median accuracy of DP-Fed-CTGAN is lower than with DP-CTGAN.

For the Obesity dataset in Figure 5.3f, all synthesizers exhibit a sharp drop in accuracy as in Figure 5.3c, since the task for this dataset is also a multiclass classification problem. The difference between TSTR and TRTR is relatively small only for AdaBoost. CTGAN, DP-CTGAN, and DP-Fed-CTGAN perform very similarly in this case, but they are again outperformed by the non-private federated methods.

All these phenomena appear in a similar form with the AUROC scores presented in Table 5.5, where only the mean value of all classifiers is reported for each dataset and synthesizer. Fed-TGAN and Fed-CTGAN achieve the highest values, while the synthesizers with DP guarantees yield lower scores and are therefore not as capable at discriminating between the classes. The results also show that our approach performs slightly better than DP-CTGAN on four of the six datasets, and is only outperformed

Dataset	Real Data	CTGAN	DP-CTGAN	Fed-TGAN	Fed-CTGAN	DP-Fed-CTGAN
Adult	0.8775	0.8279	0.7210	0.8529	0.8486	0.7526
Bank Marketing	0.9080	0.6933	0.5454	0.7484	0.7198	0.5285
Covertype	0.8863	0.6454	0.7172	0.7831	0.7611	0.6921
Cardiovascular Disease	0.7872	0.7599	0.6803	0.7543	0.7584	0.6983
Diabetes	0.8378	0.7969	0.4780	0.8068	0.7904	0.5142
Obesity	0.9293	0.5131	0.5034	0.6036	0.6096	0.5188

Table 5.5: Average AUROC scores for each dataset and SDG method.

on Bank Marketing and Covertype. The data generated with DP-CTGAN for the Diabetes dataset even lead to an average AUROC score that is inferior to the one of a random guess classifier. This could be due to the fact that the class distributions of the datasets generated by DP-CTGAN differ significantly from the class distribution of the training dataset. The strength of the AUROC metric is particularly apparent for Bank Marketing, where it is difficult to judge the various synthesizers based on the accuracy. For such heavily imbalanced datasets, evaluating utility with the AUPRC scores from Table 5.6 is even more informative, as this metric completely ignores the TNs. However, for balanced datasets, the AUPRC values are often too pessimistic.

Dataset	Real Data	CTGAN	DP-CTGAN	Fed-TGAN	Fed-CTGAN	DP-Fed-CTGAN
Adult	0.6945	0.5881	0.4403	0.6341	0.6261	0.4567
Bank Marketing	0.5629	0.2542	0.1511	0.3225	0.3010	0.1398
Covertype	0.8397	0.4846	0.5611	0.6306	0.6028	0.5335
Cardiovascular Disease	0.7633	0.7323	0.6600	0.7334	0.7386	0.6840
Diabetes	0.7289	0.6683	0.3664	0.6946	0.6723	0.3945
Obesity	0.7811	0.1757	0.1669	0.2264	0.2326	0.1763

Table 5.6: Average AUPRC scores for each dataset and SDG method.

5.2.2 Fidelity

With regard to fidelity, we first analyze how precisely the different synthesizers capture the distributions of the values of discrete and continuous columns, respectively. The average JSD between the distributions of the discrete attribute values of the real dataset and the generated datasets are reported in Table 5.7 for all methods. As already seen with the utility, the non-private federated GANs perform best. The distances are so similar that no obvious advantage for Fed-TGAN or Fed-CTGAN can be determined. With the exception of the Covertype dataset, CTGAN follows this group, while the statistical distances of the methods with DP guarantees are noticeably higher. This loss in fidelity is to be expected because of the addition of random noise on the gradients with each update of the model parameters. In the case of the Covertype dataset, DP-CTGAN and DP-Fed-CTGAN achieve a higher fidelity than CTGAN for discrete attributes. Furthermore, the JSD scores of DP-Fed-CTGAN are lower than those of DP-CTGAN for five of the six datasets.

Dataset	**CTGAN**	**DP-CTGAN**	**Fed-TGAN**	**Fed-CTGAN**	**DP-Fed-CTGAN**
Adult	0.1128	0.3229	0.0537	0.0540	0.3038
Bank Marketing	0.0855	0.2959	0.0448	0.0462	0.2639
Covertype	0.0515	0.0481	0.0223	0.0198	0.0290
Cardiovascular Disease	0.0828	0.0837	0.0511	0.0557	0.1133
Diabetes	0.1528	0.4026	0.0728	0.0765	0.2371
Obesity	0.0877	0.3362	0.0686	0.0549	0.2947

Table 5.7: Average JSD of discrete columns for each dataset and SDG method.

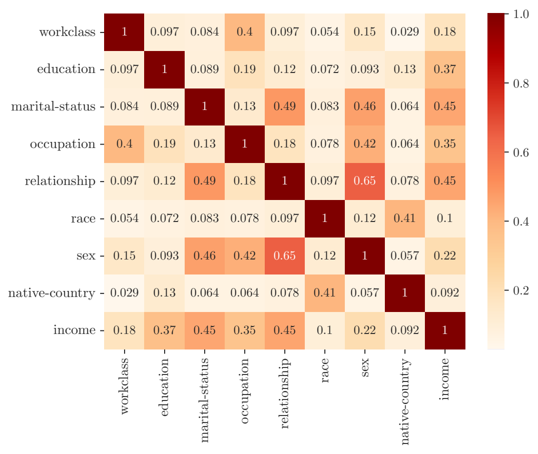
For continuous columns, a similar pattern emerges with respect to the WD metric listed in Table 5.8. The distances for Fed-TGAN and Fed-CTGAN are significantly smaller than in the centralized setting with CTGAN, which is outperformed by DP-Fed-CTGAN only for Covertype. In addition, DP-Fed-CTGAN yields a higher fidelity than DP-CTGAN with the exception of the Diabetes dataset.

Dataset	**CTGAN**	**DP-CTGAN**	**Fed-TGAN**	**Fed-CTGAN**	**DP-Fed-CTGAN**
Adult	0.0170	0.0962	0.0095	0.0140	0.0509
Bank Marketing	0.0094	0.0489	0.0062	0.0063	0.0262
Covertype	0.0262	0.0416	0.0115	0.0118	0.0182
Cardiovascular Disease	0.0087	0.0284	0.0046	0.0049	0.0253
Diabetes	0.0492	0.1829	0.0412	0.0257	0.2352
Obesity	0.1025	0.2505	0.0731	0.0573	0.2442

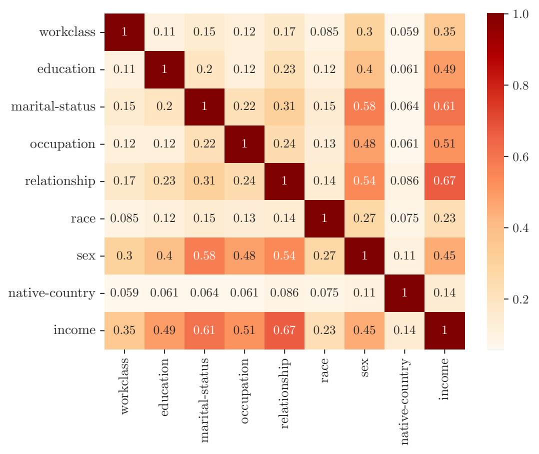
Table 5.8: Average WD of continuous columns for each dataset and SDG method.

Next, we examine the extent to which the different SDG methods preserve dependencies between the columns that exist in the training dataset. For this purpose, as shown in Figure 5.4, we visualize Cramér's V as a measure of association for each pair of discrete attributes. We selected the Adult dataset as an example and refer to the Juypter notebooks in the GitLab repository for the remaining datasets. Based on the associations between the variables in the real dataset plotted in Figure 5.4a, we observe the smallest deviations for Fed-TGAN, Fed-CTGAN, and DP-Fed-CTGAN. With CTGAN, the deviations from the real dataset are somewhat larger, while the data generated with DP-CTGAN loses most of the relations between the variables. This of course also has a significant impact on the utility of the synthetic data depending on the downstream task.

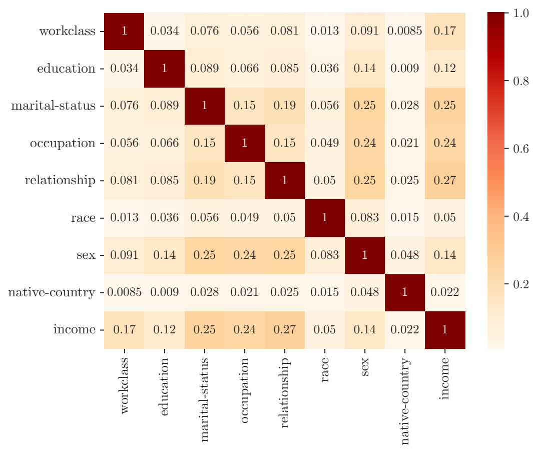
Figure 5.5, on the other hand, compares the average Pearson correlation between



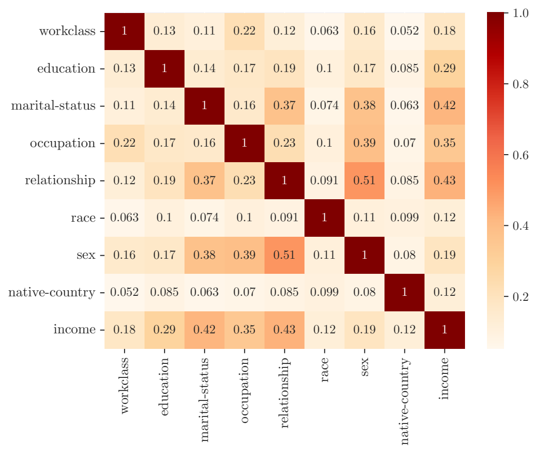
(a) Real Data



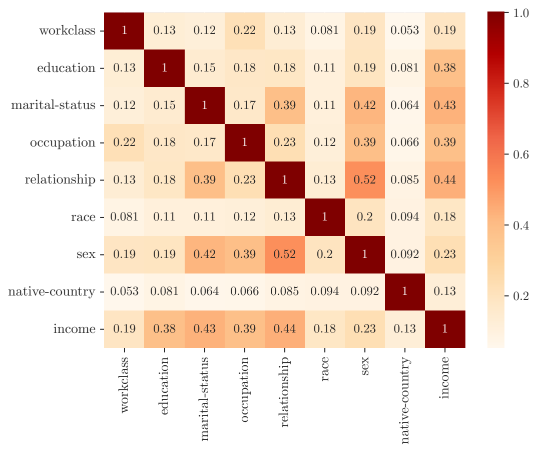
(b) CTGAN



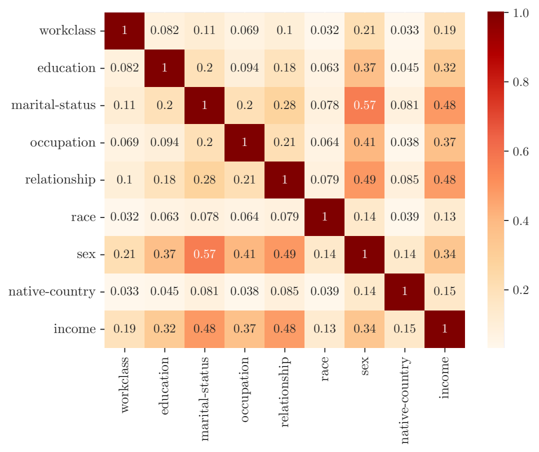
(c) DP-CTGAN



(d) Fed-TGAN

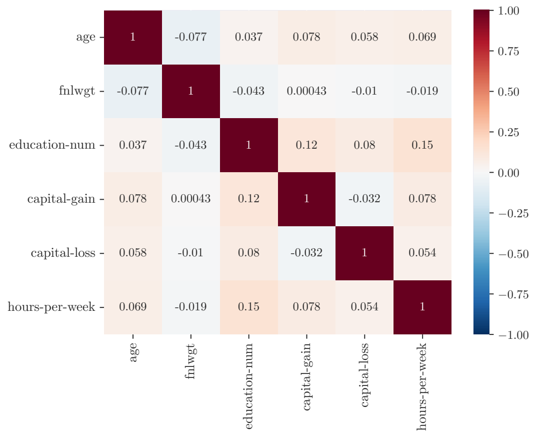


(e) Fed-CTGAN

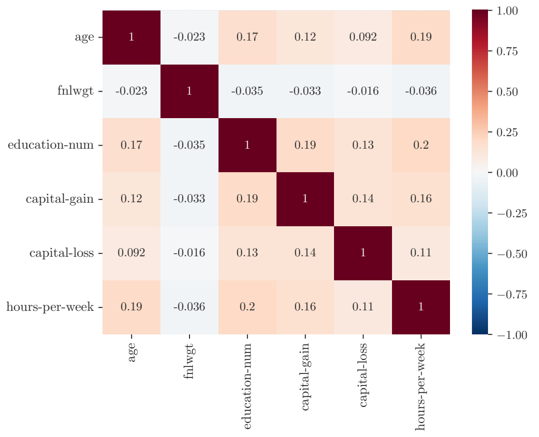


(f) DP-Fed-CTGAN

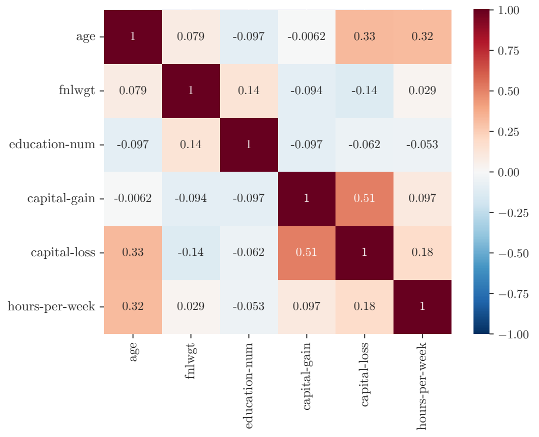
Figure 5.4: Comparison of the average Cramér's V association for discrete columns of the Adult dataset.



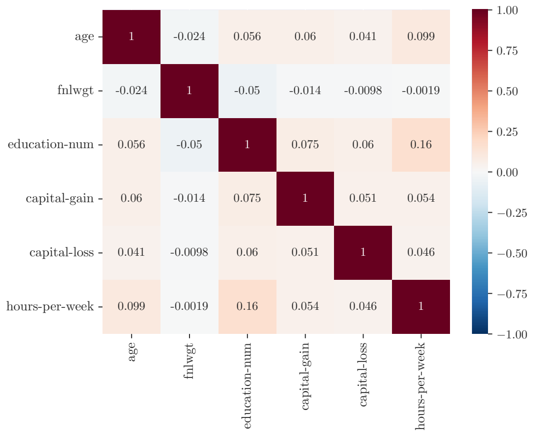
(a) Real Data



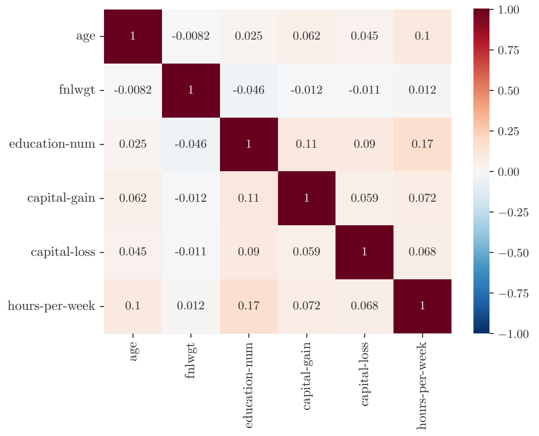
(b) CTGAN



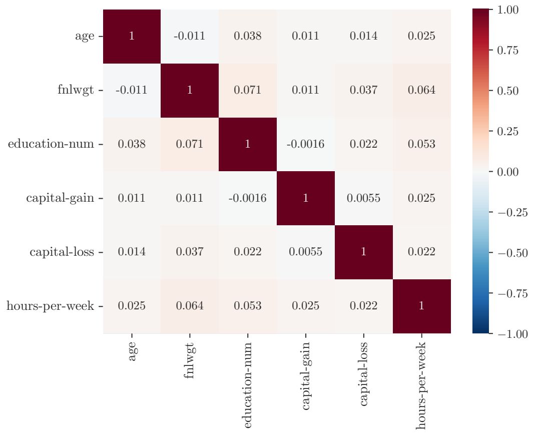
(c) DP-CTGAN



(d) Fed-TGAN



(e) Fed-CTGAN



(f) DP-Fed-CTGAN

Figure 5.5: Comparison of the average Pearson correlation for continuous columns of the Adult dataset.

all continuous columns. The lowest deviations from the correlation coefficients of the real dataset are achieved by Fed-TGAN and Fed-CTGAN. For DP-Fed-CTGAN, the correlations are predominantly weaker compared to the real data, while the correlations between the attributes increase noticeably for CTGAN. In contrast, for DP-CTGAN originally uncorrelated attributes are positively correlated, while in the real data positively correlated attributes are uncorrelated or even weakly negatively correlated in the synthetic datasets.

5.2.3 Performance for Non-IID Data

Given that our approach, in contrast to Fed-TGAN, intentionally avoids reporting the local frequencies of all discrete column values to the server, we now investigate in more detail whether this leads to sacrifices in terms of utility or fidelity. On the one hand, the absence of this information has the consequence that the conditional vectors have to be retrieved from the clients and cannot be generated on the server. Nevertheless, this should not have any other implications besides a slightly higher duration for each FL round due to additional requests over the network. On the other hand, the source for potential drawbacks of our method is that Fed-TGAN takes the frequencies of the categories of each client into account when determining the weights that are needed to aggregate the model parameters of the clients on the server, while in DP-Fed-CTGAN the weights are determined only by the local dataset sizes. The authors consider their "similarity aware weighting strategy" to be particularly beneficial in the scenario of non-IID distributed data across the clients [Zha+21].

To determine the impact of the weighting strategies for the unfavorable non-IID scenario, we selected the Cardiovascular Disease dataset and assigned all samples of the negative class, which account for 50 percent of the dataset, to one client and partitioned the positive samples between the two remaining clients. Table 5.9 shows the results of this experiment. Since a direct comparison between synthesizers with and without DP guarantees does not permit a fair assessment, we will focus on Fed-TGAN and Fed-CTGAN. First, we see that both synthesizers have significantly worse utility and fidelity than in our previous scenario, where we randomly distributed the same amount of samples among the clients. However, it can also be clearly observed that Fed-CTGAN consistently performs better in terms of utility metrics as well as in the dimension of fidelity. Moreover, DP-Fed-CTGAN achieves a marginally better fidelity than in the previous analyses and a lower WD than Fed-TGAN, while the utility decreases. Thus, we can conclude that incorporating the frequencies of the categories into the weight

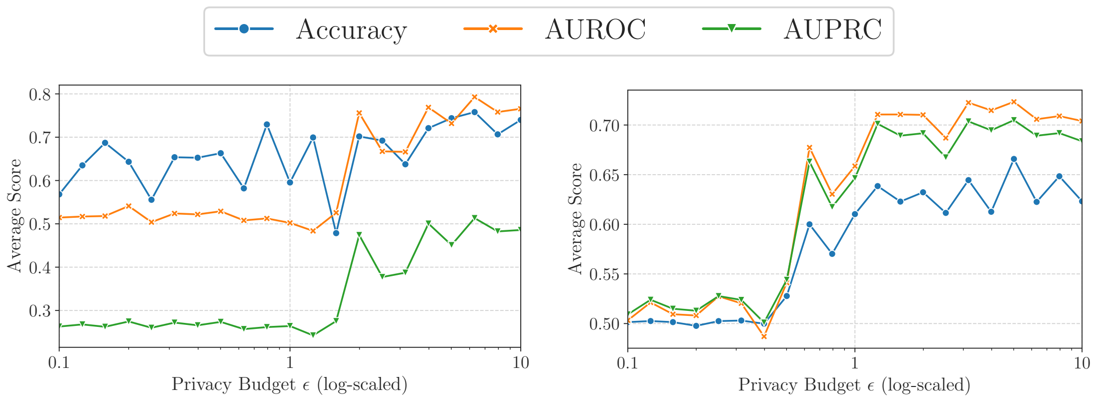
Metric	Fed-TGAN	Fed-CTGAN	DP-Fed-CTGAN
Accuracy	0.4992	0.5077	0.5098
AUROC	0.5529	0.6014	0.5322
AUPRC	0.5440	0.5861	0.5315
JSD	0.1315	0.0720	0.0905
WD	0.0067	0.0060	0.0125

Table 5.9: Average utility and fidelity metrics for the Cardiovascular Disease dataset in the non-IID scenario.

calculation actually leads to a worse performance, at least for this dataset, while at the same time our approach reduces the attack surface as described in Section 4.1.

5.2.4 Performance Under Varying Privacy Budget

In this experiment, we evaluate the utility and fidelity of DP-Fed-CTGAN for different values of the privacy parameter ε. Since for larger ε values each record is allowed to have a larger impact on the model parameters and thus less noise needs to be added in the training process, we also anticipate a better quality of the synthetic data with increasing privacy budget.

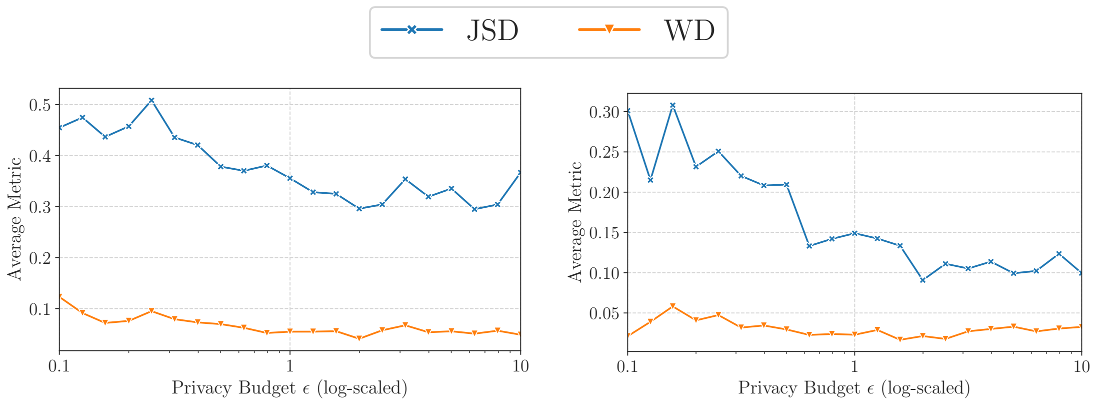


(a) Utility scores for Adult **(b)** Utility scores for Cardiovascular Disease

Figure 5.6: Influence of the privacy budget on the utility of DP-Fed-CTGAN for the Adult and Cardiovascular Disease datasets.

Figure 5.6 shows the effect of the privacy budget on the utility using the Adult and Cardiovascular Disease datasets as an example. We scaled ε logarithmically between the values 0.1 and 10 that are often found in the literature. It can be observed in Figure 5.6a that the AUROC and AUPRC scores remain constantly low up to an $\varepsilon \approx 2$ and then show a large improvement. Subsequently, there is a slight upward trend, albeit with fluctuations. The accuracy improves only marginally with increasing privacy budget, but the variability decreases. For the Cardiovascular Disease dataset in Figure 5.6b, the rapid utility improvement occurs for $\varepsilon \approx 0.6$. Here, this also applies to the accuracy scores.

Figure 5.7 illustrates the results for the fidelity. For the Adult dataset in Figure 5.7a, the JSD and WD decrease up to $\varepsilon \approx 2$. For larger values, there is no real improvement and outliers with worse scores are more frequent. The relations between the attributes, for which the plots can only be found in the corresponding Jupyter notebook due to space constraints, are captured much better by DP-Fed-CTGAN starting at $\varepsilon \approx 2$ and change only slightly thereafter. For the Cardiovascular Disease dataset in Figure 5.7b, the distances decrease significantly up to $\varepsilon \approx 0.6$, after which there is only a slight improvement up to $\varepsilon \approx 2$. However, the ability of DP-Fed-CTGAN to learn the relations between the columns improves significantly between 0.6 and 2.



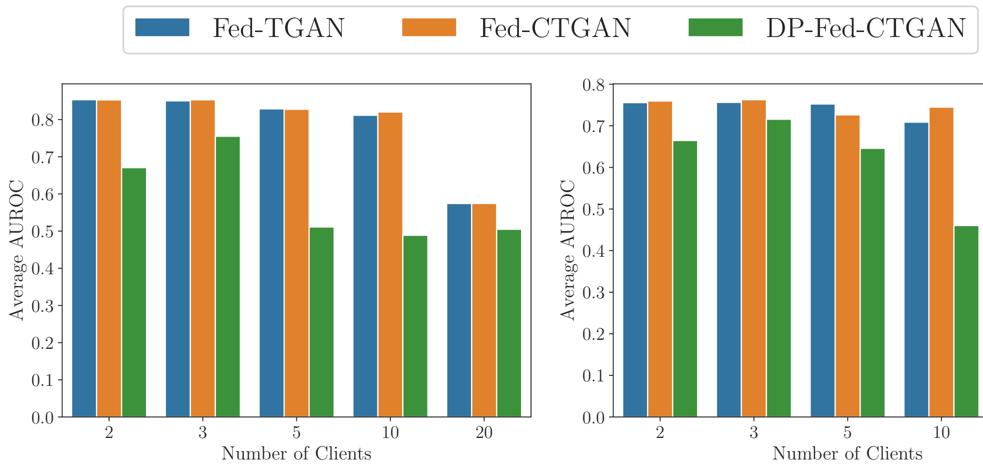
(a) Fidelity metrics for Adult **(b)** Fidelity metrics for Cardiovascular Disease

Figure 5.7: Influence of the privacy budget on the fidelity of DP-Fed-CTGAN for the Adult and Cardiovascular Disease datasets.

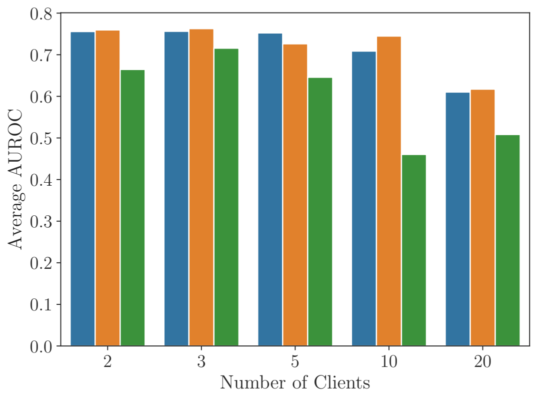
5.2.5 Performance for Different Number of Clients

In the last experiment, we analyze the effects of increasing the number of clients, while the total amount of training data available to all clients remains unchanged. This experiment simulates a realistic setting where individual healthcare providers have limited patient data. Understanding how the synthesizers perform under such constraints is essential for assessing their applicability in real-world settings.

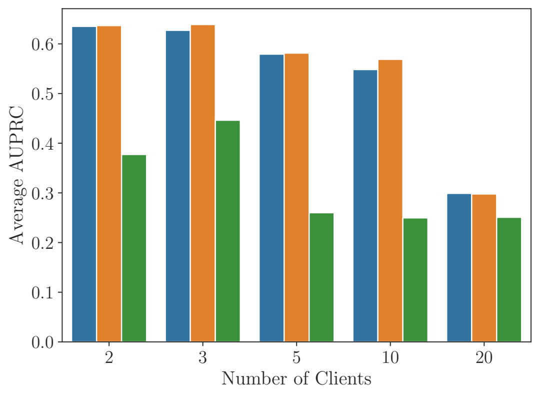
Figure 5.8 shows the results in terms of AUROC and AUPRC scores for the Adult and Cardiovascular Disease datasets. In Figure 5.8a we can observe that, as expected, the utility decreases after a certain number of clients, since at some point insufficient information is available for the GANs to learn the patterns in the data. With smaller datasets, there is also the likelihood of increased variability in the data, so the local models may have a poorer ability to generalize. It can be seen that this phenomenon occurs earlier for DP-Fed-CTGAN than for the non-private SDG methods. This is most likely due to the fact that for smaller datasets more noise has to be added during the



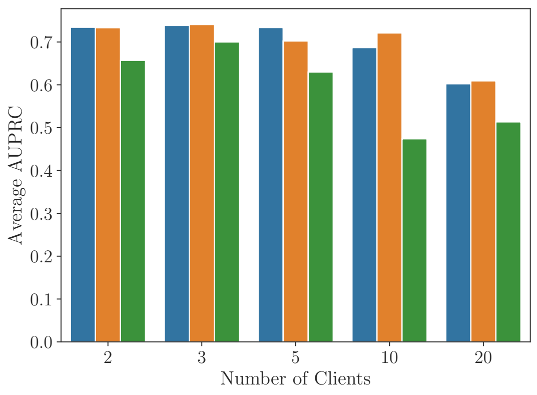
(a) AUROC score for Adult



(b) AUROC score for Cardiovascular Disease



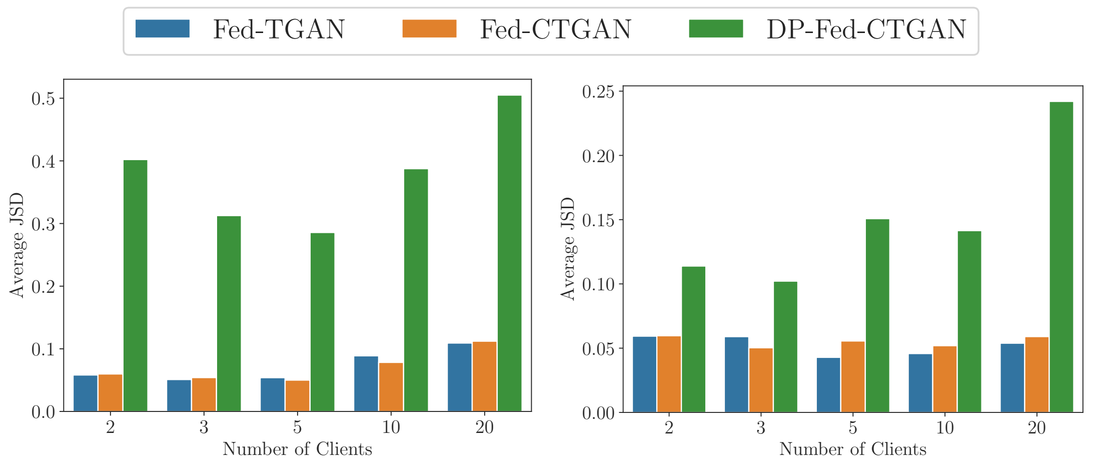
(c) AUPRC score for Adult



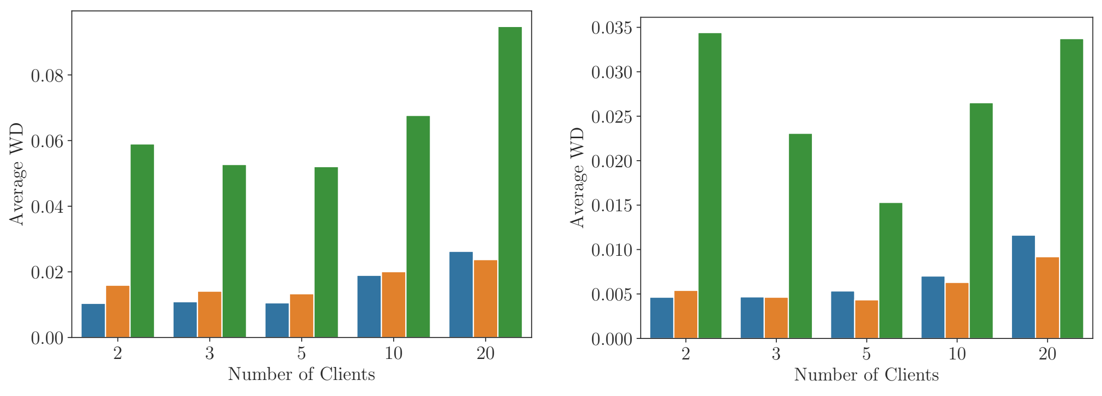
(d) AUPRC score for Cardiovascular Disease

Figure 5.8: Comparison of the utility with varying number of clients for the Adult and Cardiovascular Disease datasets.

training process to obtain the same DP guarantees. In the case of the Cardiovascular Disease dataset in Figure 5.8b, the utility decrease at 20 clients is smaller for Fed-TGAN and Fed-CTGAN, while the decline for DP-Fed-CTGAN appears later here at 10 clients. The AUPRC scores support our analyses.



(a) JSD for Adult **(b)** JSD for Cardiovascular Disease



(c) WD for Adult **(d)** WD for Cardiovascular Disease

Figure 5.9: Comparison of the fidelity with varying number of clients for the Adult and Cardiovascular Disease datasets.

Regarding the fidelity for the Adult dataset, Figure 5.9a shows a substantial increase of the JSD for the non-private GANs starting from 10 clients. In contrast, for DP-Fed-CTGAN there is initially a downward trend between two and five clients before the JSD rises noticeably. The WD in Figure 5.9c exhibits the same behavior. For the Cardiovascular Disease dataset in Figure 5.9b, an increase in JSD with growing number of clients can be observed only for DP-Fed-CTGAN. However, with respect to the WD in Figure 5.9d, a decrease in fidelity from 10 clients onwards can be identified for Fed-TGAN and Fed-CTGAN, while DP-Fed-CTGAN does not show a clear pattern.

5.3 Discussion

Even though the differentially private methods are unsurprisingly accompanied by an overall loss in utility, we observe that DP-Fed-CTGAN can keep up with the centralized setting represented by DP-CTGAN in the dimension of utility and even achieves a higher AUROC score on four of the six datasets. Thus, we conclude that the use of FL under the same DP guarantees does not incur additional costs in the form of a lower utility. At the same time, the use of FL has significant benefits, especially in the healthcare sector. First, individual medical facilities may have insufficient patient data to train a GAN that generates synthetic data of satisfactory quality. On the other hand, the local training datasets do not have to leave the individual facilities, which in the context of a medical data donation can both increase patient acceptance and facilitate adherence to internal compliance rules as well as privacy laws. Further, the results of the utility evaluation show that the non-private federated methods, Fed-TGAN and Fed-CTGAN, perform better than the centralized CTGAN. We hypothesize that FL acts as a regularization and mitigates undesired effects that may arise during training of GANs (see Section 2.2.2.4). Since there are also no significant differences between Fed-TGAN and Fed-CTGAN, our approach of minimizing the amount of information that clients have to share about their local training dataset does not seem to have a negative impact on the utility.

With regard to fidelity, the overall picture is similar. However, the JSD and WD scores of DP-Fed-CTGAN are even mostly lower than those of DP-CTGAN. DP-Fed-CTGAN actually manages to preserve the relations between the columns better than CTGAN for the most part.

In addition, the results for the non-IID scenario show that weighting the client updates without the information about the frequencies of each category leads to no loss in utility or fidelity, even under adverse conditions. In fact, our naive weighting procedure based on the local dataset sizes outperforms the weighting scheme of Fed-TGAN, which takes into account the differences in the distribution of the clients' attribute values.

In terms of the privacy budget, it seems that for each dataset a value of the parameter ε exists above which a saturation can be observed, i.e., no further improvement in utility and fidelity is noticeable. Apart from the dataset size, possible factors influencing the threshold at which this saturation occurs are the number and type of the columns as well as the class distribution. Therefore, we believe that a custom trade-off between privacy, utility, and fidelity is necessary for each dataset.

Our investigation regarding the effects of scaling the number of clients for a constant total number of samples suggests that a loss in utility and privacy occurs when the amount of training data per client drops below a certain threshold. In the case of the differentially private synthesizers, it must also be considered that a stronger gradient perturbation is necessary for smaller datasets to guarantee a certain privacy level.

Although our evaluation demonstrated the general applicability of DP-Fed-CTGAN in the scenario of a privacy-friendly medical data donation, it is important to acknowledge the limitations of our approach and findings. First, for some datasets there are significant differences in the utility achieved by the different classifiers, such that average scores are potentially misleading. In a real-world setting, the classifier performing best would be used with optimized hyperparameters. Additionally, we only considered three commonly used machine learning datasets and three datasets from the healthcare domain, so we cannot generalize our conclusions to datasets that have fundamentally different characteristics. However, there are very few publicly available datasets containing medical data, which hopefully will change in the future, also due to the use of synthetic data generators. Furthermore, we changed the hyperparameters only slightly from the default values of CTGAN, hence future work should investigate the performance with systematic tuning of the hyperparameters, e.g., by using grid search. In addition, there are other aspects that should be examined, such as the limits regarding the scalability of our approach as more and more clients participate and the total number of training samples increases. Continued research is also required on metrics that assist with the trade-off between the dimensions of privacy, utility, and fidelity. While the fidelity metrics used in the literature and also in this thesis permit a meaningful comparison of different SDG methods, they do not sufficiently take into account the fact that a very high fidelity is not necessarily desirable from a privacy perspective. Here, use case-specific metrics could also be a solution, even if the design of such metrics is time-consuming. Further weaknesses and areas for improvement can be identified by evaluating DP-Fed-CTGAN in a real-world setting.

6 Conclusion

To conclude this thesis, we summarize our approach and findings in Section 6.1 and provide an outlook for future work in Section 6.2.

6.1 Summary

This master's thesis addressed the challenge of leveraging FL for the generation of synthetic tabular data in the context of a privacy-friendly medical data donation. To achieve this, we proposed a novel approach called DP-Fed-CTGAN that provides DP guarantees and minimizes the amount of information individual clients have to share about their local training datasets during the FL process. We implemented DP-Fed-CTGAN with state-of-the-art frameworks, more specifically the CTGAN architecture for SDG, Flower for FL, and Opacus for incorporating DP.

We conducted a comprehensive evaluation of our open-source implementation of DP-Fed-CTGAN using both medical and common machine learning datasets. For this, we considered various utility and fidelity metrics to assess the quality of the generated synthetic data. The results of our utility evaluation showed that DP-Fed-CTGAN performs on par with the centralized setting represented by DP-CTGAN. In the dimension of fidelity, DP-Fed-CTGAN mostly outperforms DP-CTGAN in terms of statistical distances and even outperforms the non-private CTGAN for the most part in preserving the relations between columns. By comparing a non-private version of our approach with Fed-TGAN, we were also able to show that not reporting the frequencies of the discrete column values for weighting the client updates has no drawbacks even in the case of non-IID distributed data between the clients. We also investigated the impact of the privacy budget and the number of clients on the performance of DP-Fed-CTGAN. Our findings highlight the need for a custom trade-off between privacy, utility, and fidelity for each dataset and indicate that a sufficient amount of training data per client is crucial to avoid a reduction in utility and fidelity.

Considering the scenario of a medical data donation, we showed that DP-Fed-CTGAN can generate synthetic data without a loss in utility or fidelity compared to the centralized setting represented by DP-CTGAN, while not requiring participating facilities to share local patient data as with DP-CTGAN. In summary, we demonstrated that the combination of DP, FL, and SDG is a promising solution for generating high-quality synthetic data with mathematically rigorous privacy guarantees for medical research purposes that simultaneously mitigates privacy and security concerns when compared to a central data repository.

6.2 Future Work

While this thesis has demonstrated the general applicability of DP-Fed-CTGAN for the scenario of a privacy-preserving donation of medical data, the evaluation of the performance in a real-world setting is still an open question. Future work should also explore the effectiveness of DP-Fed-CTGAN in other domains, where privacy concerns and legal requirements can be addressed by the use of FL for the generation of differentially private synthetic data.

Moreover, protection against stronger threat models can be further enhanced by incorporating MPC techniques. Specifically, with HE each client encrypts the model parameters before sending them to the server. Then, the server performs the aggregation directly on the encrypted client parameters without prior decryption. In addition, TEEs can be utilized for both the local training on the clients and the server-side aggregation. However, there are currently significant challenges when using TEEs due to limited memory size and lack of access to GPUs.

Another area for future work is adapting our approach to hybrid FL, which is also known as federated transfer learning. In this case, the local datasets not only differ in sample space as in horizontal FL, but also in feature space. Hybrid FL aligns well with the healthcare domain, as different healthcare providers not only treat different patients, but also perform different examinations or often represent the examination results non-uniformly.

Last, personal devices such as smartphones could serve as clients in the FL scheme. Insured persons could thus decide for themselves which data from an EHR or readings from a smartwatch are donated for research purposes, thereby empowering them to actively contribute to medical advancements while maintaining control over their data. For this vision to become reality, several aspects need to be investigated in future work. On the one hand, there are significant limitations with personal devices, e.g., in terms of computing power and available bandwidth. On the other hand, the federated training of a CTGAN model with a large number of small and heterogeneous datasets may lead to poor convergence, especially when incorporating DP. Additionally, client-level DP is a more suitable privacy guarantee than record-level DP for this setting, meaning the worst-case privacy loss is bounded with respect to all samples of a single individual.

Bibliography

[Aba+16] Martin Abadi et al. “Deep Learning with Differential Privacy.” In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS ’16*. ACM, Oct. 2016, pp. 308–318. DOI: 10.1145/2976749.2978318.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein Generative Adversarial Networks.” In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 214–223. URL: https://proceedings.mlr.press/v70/arjovsky17a.html.

[Asl+23] Aycan Aslan et al. “At What Price? Exploring the Potential and Challenges of Differentially Private Machine Learning for Healthcare.” In: *Proceedings of the 56th Hawaii International Conference on System Sciences*. 2023, pp. 3277–3286. DOI: 10125/103034.

[Aso+21] Shahab Asoodeh et al. “Three Variants of Differential Privacy: Lossless Conversion and Applications.” In: *IEEE Journal on Selected Areas in Information Theory* 2.1 (Mar. 2021), pp. 208–222. DOI: 10.1109/jsait.2021.3054692.

[Aug+20] Sean Augenstein et al. “Generative Models for Effective ML on Private, Decentralized Datasets.” In: *8th International Conference on Learning Representations (ICLR 2020)*. 2020. URL: https://openreview.net/forum?id=SJgaRA4FPH.

[Beu+20] Daniel J. Beutel et al. *Flower: A Friendly Federated Learning Research Framework*. 2020. DOI: 10.48550/ARXIV.2007.14390.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006, p. 758. ISBN: 978-1-4939-3843-8.

[BK96] Barry Becker and Ronny Kohavi. *Adult*. 1996. DOI: 10.24432/C5XW20.

[Bla98] Jock Blackard. *Covertype*. 1998. DOI: 10.24432/C50K5N.

[BLC19] Mrinal Kanti Baowaly, Chao-Lin Liu, and Kuan-Ta Chen. “Realistic Data Synthesis Using Enhanced Generative Adversarial Networks.” In: *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, June 2019, pp. 289–292. DOI: 10.1109/aike.2019.00057.

[BM21] Nader Bouacida and Prasant Mohapatra. “Vulnerabilities in Federated Learning.” In: *IEEE Access* 9 (2021), pp. 63229–63249. DOI: 10.1109/access.2021.3075203.

[Bra97] Andrew P. Bradley. “The use of the area under the ROC curve in the evaluation of machine learning algorithms.” In: *Pattern Recognition* 30.7 (July 1997), pp. 1145–1159. DOI: 10.1016/s0031-3203(96)00142-2.

[BS16] Mark Bun and Thomas Steinke. *Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds*. 2016. DOI: 10.48550/ARXIV.1605.02065.

[Cho+17] Edward Choi et al. “Generating Multi-label Discrete Patient Records using Generative Adversarial Networks.” In: *Proceedings of the 2nd Machine Learning for Healthcare Conference*. Ed. by Finale Doshi-Velez et al. Vol. 68. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 286–305. URL: https://proceedings.mlr.press/v68/choi17a.html.

[Cin+21] Lucas Pinheiro Cinelli et al. “Variational Autoencoder.” In: *Variational Methods for Machine Learning with Applications to Deep Networks*. Springer International Publishing, 2021, pp. 111–149. DOI: 10.1007/978-3-030-70679-1_5.

[Cre+18] Antonia Creswell et al. “Generative Adversarial Networks: An Overview.” In: *IEEE Signal Processing Magazine* 35.1 (Jan. 2018), pp. 53–65. DOI: 10.1109/msp.2017.2765202.

[CWH20] Xiangyi Chen, Steven Z. Wu, and Mingyi Hong. “Understanding Gradient Clipping in Private SGD: A Geometric Perspective.” In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020. URL: https://proceedings.neurips.cc/paper/2020/file/9ecff5455677b38d19f49ce658ef0608-Paper.pdf.

[Dal77] Tore Dalenius. “Towards a Methodology for Statistical Disclosure Control.” In: *Statistik Tidskrift* 15 (1977), pp. 429–444.

[DE13] Fida K. Dankar and Khaled El Emam. “Practicing Differential Privacy in Health Care: A Review.” In: *Transactions on Data Privacy* 6.1 (Apr. 2013), pp. 35–67. URL: `https://www.tdp.cat/issues11/tdp.a129a13.pdf`.

[DI21] Fida K. Dankar and Mahmoud Ibrahim. “Fake It Till You Make It: Guidelines for Effective Synthetic Data Generation.” In: *Applied Sciences* 11.5 (Feb. 2021), p. 2158. DOI: `10.3390/app11052158`.

[DPT17] Differential Privacy Team, Apple. *Learning with Privacy at Scale*. Dec. 2017. URL: `https://docs-assets.developer.apple.com/ml-research/papers/learning-with-privacy-at-scale.pdf`.

[DR13] Cynthia Dwork and Aaron Roth. “The Algorithmic Foundations of Differential Privacy.” In: *Foundations and Trends® in Theoretical Computer Science* 9.3-4 (2013), pp. 211–407. DOI: `10.1561/0400000042`.

[Dro+17] Brian C. Drolet et al. “Electronic Communication of Protected Health Information: Privacy, Security, and HIPAA Compliance.” In: *The Journal of Hand Surgery* 42.6 (June 2017), pp. 411–416. DOI: `10.1016/j.jhsa.2017.03.023`.

[Dua+22] Shaoming Duan et al. “HT-Fed-GAN: Federated Generative Model for Decentralized Tabular Data Synthesis.” In: *Entropy* 25.1 (Dec. 2022), p. 88. DOI: `10.3390/e25010088`.

[Dwo+06] Cynthia Dwork et al. “Calibrating Noise to Sensitivity in Private Data Analysis.” In: *Theory of Cryptography*. Springer Berlin Heidelberg, 2006, pp. 265–284. DOI: `10.1007/11681878_14`.

[Dwo06] Cynthia Dwork. “Differential Privacy.” In: *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2006, pp. 1–12. DOI: `10.1007/11787006_1`.

[EHR17] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. *Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs*. 2017. DOI: `10.48550/ARXIV.1706.02633`.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. “RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response.” In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS ’14*. ACM, Nov. 2014, pp. 1054–1067. DOI: `10.1145/2660267.2660348`.

[Fan20] Liyue Fan. “A Survey of Differentially Private Generative Adversarial Networks.” In: *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*. 2020. URL: https://webpages.charlotte.edu/lfan4/pdf/PPAI20.pdf.

[FDK22] Mei Ling Fang, Devendra Singh Dhami, and Kristian Kersting. “DP-CTGAN: Differentially Private Medical Data Generation Using CTGANs.” In: *Artificial Intelligence in Medicine*. Springer International Publishing, 2022, pp. 178–188. DOI: 10.1007/978-3-031-09342-5_17.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. “Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures.” In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS ’15*. ACM, Oct. 2015, pp. 1322–1333. DOI: 10.1145/2810103.2813677.

[Fun+10] Benjamin C. M. Fung et al. “Privacy-Preserving Data Publishing: A Survey of Recent Developments.” In: *ACM Computing Surveys* 42.4 (June 2010), 14:1–14:53. DOI: 10.1145/1749603.1749605.

[FV22] Alvaro Figueira and Bruno Vaz. “Survey on Synthetic Data Generation, Evaluation Methods and GANs.” In: *Mathematics* 10.15 (Aug. 2022), p. 2733. DOI: 10.3390/math10152733.

[Goo+14] Ian Goodfellow et al. “Generative Adversarial Nets.” In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[Gue+20] Rachid Guerraoui et al. “FeGAN: Scaling Distributed GANs.” In: *Proceedings of the 21st International Middleware Conference*. ACM, Dec. 2020, pp. 193–206. DOI: 10.1145/3423211.3425688.

[Gui+21] Jie Gui et al. “A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications.” In: *IEEE Transactions on Knowledge and Data Engineering* (2021). DOI: 10.1109/tkde.2021.3130191.

[Gul+17] Ishaan Gulrajani et al. “Improved Training of Wasserstein GANs.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.

neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf.

[Her+23] Mikel Hernadez et al. “Synthetic Tabular Data Evaluation in the Health Domain Covering Resemblance, Utility, and Privacy Dimensions.” In: *Methods of Information in Medicine* (Jan. 2023). DOI: 10.1055/s-0042-1760247.

[HMS19] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. “MD-GAN: Multi-Discriminator Generative Adversarial Networks for Distributed Datasets.” In: *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS).* IEEE, May 2019, pp. 866–877. DOI: 10.1109/ipdps.2019.00095.

[Ink21] Becky Inkster. *Cybersecurity: A Critical Priority for Digital Mental Health.* Aug. 2021. DOI: 10.31234/osf.io/p9u3g.

[ISH22] Peter Ihle, Katharina Schneider, and Steffen Heß. *Das Forschungsdatenzentrum Gesundheit – Routinedaten der Gesetzlichen Krankenversicherung für die Gesundheits- und Versorgungsforschung.* 2022. DOI: 10.30433/GWA2022-32.

[JE19] Bargav Jayaraman and David Evans. “Evaluating Differentially Private Machine Learning in Practice.” In: *Proceedings of the 28th USENIX Security Symposium.* USENIX Association, Aug. 2019, pp. 1895–1912. URL: https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman.

[JGP16] Eric Jang, Shixiang Gu, and Ben Poole. *Categorical Reparameterization with Gumbel-Softmax.* 2016. DOI: 10.48550/ARXIV.1611.01144.

[Jor+22] James Jordon et al. *Synthetic Data - what, why and how?* 2022. DOI: 10.48550/ARXIV.2205.03257.

[JYS19] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. “PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees.” In: *7th International Conference on Learning Representations (ICLR 2019).* 2019. URL: https://openreview.net/forum?id=S1zk9iRqF7.

[Kai+21] Peter Kairouz et al. “Advances and Open Problems in Federated Learning.” In: *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210. DOI: 10.1561/2200000083.

[Kau+20] Dhamanpreet Kaur et al. "Application of Bayesian networks to generate synthetic health data." In: *Journal of the American Medical Informatics Association* 28.4 (Dec. 2020), pp. 801–811. DOI: 10.1093/jamia/ocaa303.

[KB14] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2014. DOI: 10.48550/ARXIV.1412.6980.

[KOV17] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. "The Composition Theorem for Differential Privacy." In: *IEEE Transactions on Information Theory* 63.6 (June 2017), pp. 4037–4049. DOI: 10.1109/tit.2017.2685505.

[KW13] Diederik P. Kingma and Max Welling. *Auto-Encoding Variational Bayes.* 2013. DOI: 10.48550/ARXIV.1312.6114.

[LeC+89] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition." In: *Neural Computation* 1.4 (Dec. 1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[Li+21] Qinbin Li et al. "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection." In: *IEEE Transactions on Knowledge and Data Engineering* (2021). DOI: 10.1109/tkde.2021.3124599.

[Li+22] Wei Li et al. "IFL-GAN: Improved Federated Learning Generative Adversarial Network With Maximum Mean Discrepancy Model Aggregation." In: *IEEE Transactions on Neural Networks and Learning Systems* (2022), pp. 1–14. DOI: 10.1109/tnnls.2022.3167482.

[Lin+17] Zinan Lin et al. *PacGAN: The power of two samples in generative adversarial networks.* 2017. DOI: 10.48550/ARXIV.1712.04086.

[Liu+22] Xiaoyuan Liu et al. *UniFed: A Benchmark for Federated Learning Frameworks.* 2022. DOI: 10.48550/ARXIV.2207.10308.

[LKA16] Jake Lever, Martin Krzywinski, and Naomi Altman. "Classification evaluation." In: *Nature Methods* 13.8 (July 2016), pp. 603–604. DOI: 10.1038/nmeth.3945.

[LLV07] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. "t-Closeness: Privacy Beyond k-Anonymity and ℓ-Diversity." In: *2007 IEEE 23rd International Conference on Data Engineering.* IEEE, Apr. 2007, pp. 106–115. DOI: 10.1109/icde.2007.367856.

[Mac+07] Ashwin Machanavajjhala et al. "ℓ-diversity: Privacy Beyond k-Anonymity." In: *ACM Transactions on Knowledge Discovery from Data* 1.1 (Mar. 2007), p. 3. DOI: 10.1145/1217299.1217302.

[McM+17a] H. Brendan McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data." In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, Apr. 2017, pp. 1273–1282. URL: https://proceedings.mlr.press/v54/mcmahan17a.html.

[McM+17b] H. Brendan McMahan et al. *Learning Differentially Private Recurrent Language Models*. 2017. DOI: 10.48550/ARXIV.1710.06963.

[MCR14] Sérgio Moro, Paulo Cortez, and Paulo Rita. *Bank Marketing*. 2014. DOI: 10.24432/C5K306.

[McS09] Frank McSherry. "Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis." In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, June 2009, pp. 19–30. DOI: 10.1145/1559845.1559850.

[Mir17] Ilya Mironov. "Rényi Differential Privacy." In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, Aug. 2017, pp. 263–275. DOI: 10.1109/csf.2017.11.

[ML16] Shakir Mohamed and Balaji Lakshminarayanan. *Learning in Implicit Generative Models*. 2016. DOI: 10.48550/ARXIV.1610.03483.

[MO14] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. DOI: 10.48550/ARXIV.1411.1784.

[MP21] Kenneth D. Mandl and Eric D. Perakslis. "HIPAA and the Leak of 'Deidentified' EHR Data." In: *New England Journal of Medicine* (June 2021). DOI: 10.1056/nejmp2102616.

[MT07] Frank McSherry and Kunal Talwar. "Mechanism Design via Differential Privacy." In: *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*. IEEE, Oct. 2007, pp. 94–103. DOI: 10.1109/focs.2007.66.

[Mul19] Trix Mulder. “Health Apps, their Privacy Policies and the GDPR.” In: *European Journal of Law and Technology* 10.1 (June 2019). URL: http://ejlt.org/index.php/ejlt/article/view/667.

[NA21] Joseph P. Near and Chiké Abuah. *Programming Differential Privacy*. Vol. 1. 2021. URL: https://uvm-plaid.github.io/programming-dp/.

[NAC07] Mehmet Ercan Nergiz, Maurizio Atzori, and Chris Clifton. “Hiding the Presence of Individuals from Shared Databases.” In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data - SIGMOD ’07*. ACM Press, 2007, pp. 665–676. DOI: 10.1145/1247480.1247554.

[Ngu+19] Thanh Thi Nguyen et al. “Deep Learning for Deepfakes Creation and Detection: A Survey.” In: (2019). DOI: 10.48550/ARXIV.1909.11573.

[NHC22] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. “Local and Central Differential Privacy for Robustness and Privacy in Federated Learning.” In: *29th Annual Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2022. DOI: 10.14722/ndss.2022.23054.

[NS08] Arvind Narayanan and Vitaly Shmatikov. “Robust De-anonymization of Large Sparse Datasets.” In: *2008 IEEE Symposium on Security and Privacy (S&P ’08)*. IEEE, May 2008, pp. 111–125. DOI: 10.1109/sp.2008.33.

[Pan+19] Zhaoqing Pan et al. “Recent Progress on Generative Adversarial Networks (GANs): A Survey.” In: *IEEE Access* 7 (2019), pp. 36322–36333. DOI: 10.1109/access.2019.2905015.

[Pap+17] Nicolas Papernot et al. “Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data.” In: *5th International Conference on Learning Representations (ICLR 2018)*. 2017. URL: https://openreview.net/forum?id=HkwoSDPgg.

[Pap+18] Nicolas Papernot et al. “Scalable Private Learning with PATE.” In: *6th International Conference on Learning Representations (ICLR 2018)*. 2018. URL: https://openreview.net/forum?id=rkZB1XbRZ.

[RHM19] Luc Rocher, Julien M. Hendrickx, and Yves-Alexandre de Montjoye. "Estimating the success of re-identifications in incomplete datasets using generative models." In: *Nature Communications* 10.1 (July 2019). DOI: 10.1038/s41467-019-10933-3.

[Ros+20] Lucas Rosenblatt et al. *Differentially Private Synthetic Data: Applied Evaluations and Enhancements.* 2020. DOI: 10.48550/ARXIV.2011.05537.

[RSR20] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. *FedGAN: Federated Generative Adversarial Networks for Distributed Data.* 2020. DOI: 10.48550/ARXIV.2006.07228.

[Rub93] Donald B. Rubin. "Statistical disclosure limitation." In: *Journal of Official Statistics* 9.2 (1993), pp. 461–468.

[Sar+22] Tabinda Sarwar et al. "The Secondary Use of Electronic Health Records for Data Mining: Data Characteristics and Challenges." In: *ACM Computing Surveys* 55.2 (Jan. 2022), 33:1–33:40. DOI: 10.1145/3490234.

[SC21] Divya Saxena and Jiannong Cao. "Generative Adversarial Networks (GANs)." In: *ACM Computing Surveys* 54.3 (May 2021), 63:1–63:42. DOI: 10.1145/3446374.

[Sch+21] James Scheibner et al. "Revolutionizing Medical Data Sharing Using Advanced Privacy-Enhancing Technologies: Technical, Legal, and Ethical Synthesis." In: *Journal of Medical Internet Research* 23.2 (Feb. 2021), e25120. DOI: 10.2196/25120.

[SK20] Shahid Munir Shah and Rizwan Ahmed Khan. "Secondary Use of Electronic Health Record: Opportunities and Challenges." In: *IEEE Access* 8 (2020), pp. 136947–136965. DOI: 10.1109/access.2020.3011099.

[SOT22] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. "Synthetic Data – Anonymisation Groundhog Day." In: *Proceedings of the 31st USENIX Security Symposium.* USENIX Association, Aug. 2022, pp. 1451–1468. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/stadler.

[Swe02] Latanya Sweeney. "k-Anonymity: A Model for Protecting Privacy." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (Oct. 2002), pp. 557–570. DOI: 10.1142/s0218488502001648.

[Tha20] Alaa Tharwat. “Classification assessment methods.” In: *Applied Computing and Informatics* 17.1 (July 2020), pp. 168–192. DOI: 10.1016/j.aci.2018.08.003.

[Vim+12] Sabrina De Capitani di Vimercati et al. “Data Privacy: Definitions and Techniques.” In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 20.06 (Dec. 2012), pp. 793–817. DOI: 10.1142/s0218488512400247.

[Wal+17] Jason Walonoski et al. “Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record.” In: *Journal of the American Medical Informatics Association* 25.3 (Aug. 2017), pp. 230–238. DOI: 10.1093/jamia/ocx079.

[WZH17] Zhiqiang Wan, Yazhou Zhang, and Haibo He. “Variational Autoencoder Based Synthetic Data Generation for Imbalanced Learning.” In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Nov. 2017. DOI: 10.1109/ssci.2017.8285168.

[Xie+18] Liyang Xie et al. *Differentially Private Generative Adversarial Network.* 2018. DOI: 10.48550/ARXIV.1802.06739.

[Xin+20] Bangzhou Xin et al. “Private FL-GAN: Differential Privacy Synthetic Data Generation Based on Federated Learning.” In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2020, pp. 2927–2931. DOI: 10.1109/icassp40776.2020.9054559.

[Xin+22a] Bangzhou Xin et al. “Federated synthetic data generation with differential privacy.” In: *Neurocomputing* 468 (Jan. 2022), pp. 1–10. DOI: 10.1016/j.neucom.2021.10.027.

[Xin+22b] Xiaodan Xing et al. *Non-Imaging Medical Data Synthesis for Trustworthy AI: A Comprehensive Survey.* 2022. DOI: 10.48550/ARXIV.2209.09239.

[Xu+19] Lei Xu et al. “Modeling Tabular Data using Conditional GAN.” In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf.

[Xu20] Lei Xu. “Synthesizing Tabular Data using Conditional GAN.” MA thesis. Massachusetts Institute of Technology, 2020. DOI: 1721.1/128349.

[Yan+19] Qiang Yang et al. “Federated Machine Learning: Concept and Applications.” In: *ACM Transactions on Intelligent Systems and Technology* 10.2 (Jan. 2019), 12:1–12:19. DOI: 10.1145/3298981.

[YGP09] Jim Young, Patrick Graham, and Richard Penny. “Using Bayesian Networks to Create Synthetic Data.” In: *Journal of Official Statistics* 25.4 (2009), pp. 549–567.

[Yon+19] Ryo Yonetani et al. *Decentralized Learning of Generative Adversarial Networks from Non-iid Data.* 2019. DOI: 10.48550/ARXIV.1905.09684.

[You+21] Ashkan Yousefpour et al. “Opacus: User-Friendly Differential Privacy Library in PyTorch.” In: *NeurIPS 2021 Workshop Privacy in Machine Learning.* 2021. URL: https://openreview.net/forum?id=EopKEYBoI-.

[Zha+14] Jun Zhang et al. “PrivBayes: Private Data Release via Bayesian Networks.” In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data - SIGMOD ’14.* ACM, June 2014, pp. 1423–1434. DOI: 10.1145/2588555.2588573.

[Zha+21] Zilong Zhao et al. *Fed-TGAN: Federated Learning Framework for Synthesizing Tabular Data.* 2021. DOI: 10.48550/ARXIV.2108.07927.

[Zhu+21] Hangyu Zhu et al. “Federated learning on non-IID data: A survey.” In: *Neurocomputing* 465 (Nov. 2021), pp. 371–390. DOI: 10.1016/j.neucom.2021.07.098.

[Zig+20] Athanasios Zigomitros et al. “A Survey on Privacy Properties for Data Publishing of Relational Data.” In: *IEEE Access* 8 (2020), pp. 51071–51099. DOI: 10.1109/access.2020.2980235.

[ZLH19] Ligeng Zhu, Zhijian Liu, and Song Han. “Deep Leakage from Gradients.” In: *Advances in Neural Information Processing Systems.* Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf.