Differentially private synthetic tabular data generation with a generative adversarial network and privacy amplification by subsampling

Master's Thesis
University of Turku
Department of Computing
Data Analytics
2022
Valtteri Nieminen

UNIVERSITY OF TURKU
Department of Computing

VALTTERI NIEMINEN: Differentially private synthetic tabular data generation with a generative adversarial network and privacy amplification by subsampling

Master's Thesis, 87 p.
Data Analytics
August 2022

Advances in computation have created high demand for large datasets, which in turn has sparked interest in using personal data collected by different institutions for secondary purposes such as research. However, in many domains like healthcare, privacy concerns often stand in the way of sharing data for novel use.

One promising approach to making data anonymous in order to make privacy-preserving data sharing possible is to create what is called synthetic data. Synthetic data is based on real data and attempts to mimic the properties of that real data while preserving utility for different tasks and protecting the privacy of those depicted in the original dataset.

In this work, the Differential Privacy (DP) framework is adopted to train a Generative Adversarial Network (GAN) in a privacy-preserving manner to create differentially private synthetic tabular data. The quality of the synthetic data is evaluated based on its usefulness for training new models on it, the extent to which realistic sample quality is retained and the strength of privacy guarantees achieved.

The technical implementation modifies the state-of-the-art DP GAN model, the GS-WGAN by Chen, Orekondy, and Fritz [1] from the domain of images to that of tabular data. This model choice poses novel questions on whether similar privacy benefits as reported with image data can be achieved with tabular data by applying the privacy by subsampling technique to the GAN training process. The technical choices in this work also focus on theoretical synergies between the model architecture and privacy-preserving training as well as the method's usability in a real-life scenario.

The results show that the synthetic data generated preserves utility in training downstream classification models while attaining strong privacy guarantees. However, simultaneously retaining realistic sample quality proved to be difficult. The research presented in this thesis contributes to the field of differentially private synthetic data generation with GAN models by demonstrating, that the application of PABS to GAN training is an effective way to achieve stronger privacy guarantees with tabular data. The results raise important questions over whether the use of downstream classification accuracy as a metric can lead to synthetic data biased towards this specific task and whether DP synthetic data should be separately crafted for different tasks to avoid loss of utility.

Keywords: Synthetic Data, Differential Privacy, Data Sharing, Generative Adversarial Networks

Contents

List of acronyms

BCE Binary cross-entropy A.K.A log loss

DP Differential privacy

DPSGD Differentially private stochastic gradient descent

EHR Electronic health record

GAN Generative adversarial neural network

GDPR General data protection regulation of the EU

GP Gradient penalty regularization term

KL Kullback-Leibner divergence

LR Logistic regression

MLE Maximum likelihood estimation

ML Machine learning

NN Neural network

PABS Privacy amplification by subsampling

PATE Private aggregation of teacher ensembles

RDP Rényi differential privacy

ReLu Rectified linear unit activation function

SDC Statistical disclosure control

SGD Stochastic gradient descent

Tanh Hyperbolic tangent activation function

Wloss Wasserstein loss

1 Introduction

Data that can be distilled into useful information is the lifeblood of scientific progress and sharing individual-level health data for research or other *secondary use* has the potential to benefit healthcare by accelerating this progress significantly. In this thesis secondary use refers to the use of data for other than the purpose it was initially collected for. The term comes from the Finnish secondary use act of 2019 [2], which lists as secondary uses such as scientific research, development and innovation, statistics, and use of data for policy-making. This choice of terminology follows from the legal context in which data sharing is examined throughout this work; the sharing of personal data in Finland and in the European Union (EU).

Personal health data used with machine learning and data-analytics projects are often a part of *electronic health records* (EHR). In the eyes of EU law, an *electronic health record* is defined as a "comprehensive medical record or similar documentation of the past and present physical and mental state of health of an individual in electronic form ..." [3]. In practice, this comprehensiveness is not as strict but is used to refer to many different kinds of health data, such as information collected during treatment and examinations as well as related statements and doctor notes.

EHR data paired with modern machine learning methods have produced not only promises but also powerful applications and findings such as detection of Parkinson's disease [4] and mortality prediction [5]. Along with visions of future potential, such accomplishments have led to increased research interest in finding ways to make

tapping into the potential of secondary use of health data possible.

Unfortunately, EHR data is remarkable not only due to its prospects but also in terms of the privacy risks it poses to those whose sensitive information it holds. Knowledge of one's medical history, whether acquired legitimately or not, can render them vulnerable to abuse and give considerable power over an individual. Because of this, the privacy requirements of EHR data sharing are often understandably stringent, and fulfilling these requirements is often the most significant challenge [6] standing before the secondary use of health data.

There are, in general, two ways in which the secondary use of personal data is permitted by law [7]. First, the EU data protection regulations do not consider *anonymous* information to be personal information, meaning it could be shared for secondary use. While this opens up possibilities for solutions, what specifically would be anonymous enough or what the process of adequate *anonymization* would look like in the EU is left ambiguous. The General Data Protection Regulation (GDPR) directive [8] for example, gives a broad statement that data should be "...rendered anonymous in such a manner that the data subject is not or no longer identifiable".

The second possibility for data holders to share sensitive information for secondary use without fears of legal problems is to receive consent from those it concerns. In practice, this is very difficult to achieve, even in the case of small studies, making this approach borderline impossible for large datasets. It has also been shown [9] that demanding informed consent may introduce data quality issues resulting in non-representative data through selection bias caused by differing characteristics between those who consent and those who do not.

Due to these boundaries, the choice for those willing to share their data for secondary use comes down to a decision between ambiguous and practically impossible. The result of this is that different techniques attempting to make data anonymous to ensure privacy while preserving data utility have become the go-to approach to

make data sharing possible.

1.1 Traditional anonymization practices

Most of the many data anonymization solutions described in the literature surrounding anonymization can be seen as a result of work done in the subfield of statistics known as Statistical Disclosure Control (SDC), with different methods owning to this field having been used since the mid-twentieth century [10]. SDC methods aim to release data products, such as results of statistical tests or other computations that protect privacy while preserving as much *utility* as possible [11]. In this thesis, *utility* refers to the extent to which the results of, for example, models trained or statistical descriptives calculated from the anonymized data are similar to those of the original data.

According to a 2015 review on practices of anonymization and sharing of individual patient data [7], most anonymization guidelines approach the privacy risk of data by dividing variables into *direct identifiers*, and *quasi-identifiers*. *Direct identifiers* are features that solely identify a person, such as social security numbers or names. *Quasi-identifiers* can be used to indirectly identify a person together with some other information, such as level of income or postal code. This division is characteristic of the SDC approach to protecting privacy; for data to be considered anonymous, features considered to be in either of these categories must be addressed [7]. The actions traditionally taken in SDC methods to address the privacy risks of these variables are based on removing or obscuring these direct and quasi-identifying variables, and their combinations [11]. For example, these can be generalizing age or level of income values to bins, aggregating values by groups, or adding noise to the results of a query or computation (see for example [11]). In this thesis, the term *feature* which is often used in the field of machine learning instead of the term variable is solely used in the later chapters. Variable and feature are used in this

introductory chapter interchangeably due to discussing SDC-methods, where the term variable, that comes from the context of statistics is traditionally used.

Shortcomings of traditional practices

Traditional SDC methods and anonymization practices have been noticed to be insufficient to ensure privacy and have been under attack in the past decades [12] as many researchers have demonstrated that individuals in datasets thought to be anonymous can be *re-identified*. In what follows, this insufficiency is exemplified by looking at three critical weaknesses of traditional anonymization. The list is by no means complete, but it highlights crucial aspects of their failure to protect privacy in today's world and the need for new methods serving as a justification for the decision to adopt the framework of *Differential Privacy* (DP) for this work.

In the context of EHRs, a seminal demonstration of these weaknesses is the successful re-identification of individuals using anonymized hospital records and voter registration data by Latanya Sweeney in 2002 [13]. Sweeney acquired the hospital-level health data of 135 000 state employees from an insurance company, which shared data with researchers. She then went on to purchase publicly available voter registration data from the state. Using the information (year of birth, postal code, and sex) available in both datasets Sweeney was available to re-identify subjects in both. Included in the subjects whose health data was re-identified was also the governor of Massachusetts, which gives an idea of the power of her demonstration.

Sweeney's work reveals the first of the shortcomings of traditional methods: what is considered anonymous can be ambiguous. Both datasets were considered to have been anonymized in terms of identifying variables but, in truth, were not. The decision on which variables were identified as risky and how much they should be perturbed was decided based on a subjective opinion of the data holder. In other words, the problem is that this way of anonymization does not answer the question

of how anonymous some data is because there is no measure for anonymity. The lack of a measure also makes methods incomparable to others and results in claims of anonymity being difficult to show to be true or false.

In literature, attacks such as the one by Sweeney are referred to as *linkage attacks* or re-identification attacks. More generally, attacks against privacy are called *adversarial attacks* where the actor performing the attack is called an *adversary* (see for example [7]).

A critical reader might argue that in the above case, the release of the zip code and year of birth clearly shows the data is anonymized poorly and that this example is low-hanging fruit. It has, however, been shown that successful linkage can be achieved even when no clear identifiers are present. In 2006, the streaming company Netflix released data of 100 million movie ratings comprising of a made-up user id, movies the user had rated, the date of the rating at a precision of two weeks, and the grade given as an integer [14]. Two years later, in their classic work, Narayan et al. [15] showed that it took only the rating and time of that rating for three movies to uniquely identify 80% of users in the data. This means that the user has a combination of values that separates them from other observations in the dataset. The authors went on to link the data with accounts of the Internet Movie Database (IMDB) website, demonstrating that it is possible in many cases and with little uncertainty to successfully link a set of reviews to an account given only a handful of reviews [15].

The before example highlights a second critical weakness with traditional methods; as the amount of freely available information grows, so grow the opportunities for successful re-identification attacks. Traditional anonymization methods can not defend against such potential present or future *auxiliary information* an adversary might have. These problems stem from how traditional SDC methods often base their estimates of privacy risk on assumptions over the knowledge that an adver-

sary has and the strategy an adversary is using (see for ex. [11]). One of these assumptions is the previously covered focus on choosing specific variables as being privacy-risky.

In addition to the problem of auxiliary information, new hardware and software available to attackers can be listed as the third weakness of traditional methods as assumptions about the strategy of an adversary can not account for the progress in computing or new types of attacks as is the case with the linkage attacks described. In other words, the problem is that if one can not cover future developments, the situation will remain as an arms race between adversaries and developers of anonymization methods, where the defending party will always be a step behind. Due to this dynamic between SDC methods and the difficulty of protecting against new attacks, the protection given by many SDC-based anonymization methods is considered time-limited, with estimates ranging between 18-24 months [7].

1.2 Stronger defence with differential privacy

In this thesis, the weaknesses related to SDC methods and the problem of protecting privacy while maintaining utility are tackled with the theory of differential privacy [16]. In what follows, we will see how the shortcomings presented before can be addressed with DP. To summarize, these were the ambiguity of what is considered anonymous, the failure to defend against future or present auxiliary information of an adversary due to assumptions, and the failure to address adversary access to new techniques and computing power. Differential privacy addresses these problems by giving a measure of privacy loss for any randomized computational process such as an algorithm, guaranteeing a worst-case bound for this privacy loss for present or future auxiliary knowledge and guaranteeing that this bound holds for all future attacks [10] [17]. Also, the weakness of focusing on specific variables that tie the scope to a particular set of data and set of variables is eliminated by DP. This, as well

as the ability to give defense over unknown attacks and information, is due to DP having a different approach to protecting privacy in comparison to SDC methods; DP does not focus on specific variables but the difference in information that can be learned from one individual observation, as opposed to what can be learned from that observation based on other observations.

Although differential privacy builds on the same line of research as traditional anonymization methods [17], DP is not an algorithm in itself, but a mathematical definition algorithms can fulfill. This gives an advantage over methods crafted specifically for some specific data or attack, as DP can be flexibly applied to algorithms already found to work on a problem. Importantly, DP also provides a mathematically rigorous way to compare different ways of protecting privacy in data.

Differentially private synthetic data and data sharing

The approach taken in this thesis to enable private data sharing for secondary use is to create *synthetic data* with differential privacy guarantees. Giving an accurate definition of the term can be an elusive task, but one of the central ideas is that it is based on some real data and "tries to mimic" that real data as put by Rubin [18], accredited for coming up with the idea of using synthetic data to preserve privacy. It is important to separate the work that uses synthetic data to enable private data sharing from other uses; In the field of machine learning, synthetic data is used not only for this purpose, but in, for example, data augmentation and reinforcement learning [19], as well as missing value imputation [20].

In this work, differentially private synthetic data is considered to be data that attempts to preserve utility while protecting privacy and is based on real data. Unless otherwise stated, the term synthetic data is used in the context of preserving privacy. In order to avoid confusion, it is good to mention that in some work,

synthetic data without privacy guarantees, such as in the work of Rubin [21] is also claimed to be privacy-preserving, although no mathematical privacy guarantees are given. In this thesis, only synthetic data that comes with differential privacy guarantees is considered privacy-preserving. A distinction is also made between synthetic and simulated data. While synthetic data is based on real data, simulated data refers to data not directly based on real data and is of no concern to this work. To further clarify this difference, in this context, directly based means using some learning algorithm to generate data. In the case of simulated data, learning would not happen, and the process could be, for example, sampling the values of a feature from some distribution with parameters determined by domain knowledge (such as sampling values for height from a normal distribution).

The choice of synthetic data as the end-product of differentially private data sharing is by no means self-evident, nor is it the only way to enforce DP guarantees in a data-analysis task. What can be achieved with DP synthetic data can be achieved through other means also. For example, if one's goal is to calculate statistics from the synthetic dataset, an alternative would be to directly calculate the statistics from the real data but with a differentially private version of the algorithm used. In fact, synthesizing data, even without perturbations, will almost always (except for by chance coming closer to the population distribution than the sample used is) degrade utility compared to calculating the same directly from private data due to information lost in the process. Consequently, in some applications, such as those in the health domain where inaccurate inference might lead to suffering, data should not be synthesized with or without differential privacy guarantees.

Despite the possible loss of information, synthetic data has significant benefits that make it a good option for many tasks. If one could create high-quality synthetic data with adequate privacy guarantees, it would be a very convenient way of sharing data on a small as well as a large scale because it would arguably burden an

organization willing to share data less. For example, for an organization to let users make DP-calculations on some platform on sensitive data, they would need to hire someone with expertise in differential privacy to host these services. With synthetic data, no personnel would be needed to maintain it after it is generated.

Perhaps the most promising uses of this kind of synthetic data are related to the development and testing of models and teaching. For all readers, this might not sound too exciting, but for a research group or a company developing a model or a product, getting any relevant data to test their methods can be a time- and resource-consuming process that can either enable or ruin progress. For example, in Finland, applications to get data from the national data service FinData took 2-12 weeks to be processed for 75% of applications and 3-6 months for 25% of applications in 2021 [22], which does not yet even mean that one would get access to data.

1.3 Research aims

The objective of the research presented in this thesis is to create differentially private synthetic data, specifically tabular EHR data using a Generative Adversarial Network (GAN) [23]. In this thesis tabular data refers to data with mixed feature types and is not data where features relate to each other by being parts of some larger whole, for example, pixels in images or audio separated into frequencies. Tabular data, can be thought to comprise of rows and columns, where every column represents a feature and the rows are individual patients. The synthetic tabular EHR data generated in this work is based on the Kaggle Cardiovascular disease dataset [24], consisting of 12 features and 70 000 observations.

The technical approach chosen is to modify the state-of-the-art privacy-preserving GAN technique named GS-WGAN by Chen. et al [1], from the domain of images to that of tabular data. Of special interest is whether the privacy by subsampling [25] technique and the sanitization tactic that takes advantage of theoretical synergies

between the Wasserstein-loss coupled with Gradient penalty [26] and Differentially Private Stochastic Gradient Descent (DPSGD) can produce good sample quality with strong privacy guarantees not only with images but also with tabular data.

The synthetic data is evaluated on three fronts: the strength of privacy guarantees achieved, downstream modeling utility at different privacy levels, and sample quality evaluations also at different strengths of privacy guarantees. Decisions regarding the way synthetic data are evaluated focus on whether the approach could be used in a real-life setting. The downstream modeling utility task tests if synthetic data produced by this method could be used for training a classifier that would work on real data. The sample quality evaluations investigate whether this method can produce private synthetic data with realistic samples and be useful for applications where realism is important, such as teaching.

The technical approach was also chosen with usability in mind: first, a GAN network is used as it can be, in comparison to many other DP synthetization methods, quickly changed to work with other datasets and different data types. Second, the chosen method's privacy guarantee calculation is dataset independent. This means that no privacy-parameter tuning is needed. Third, this thesis refrains from extensive optimization of model-related hyperparameters, because for a method to be used by, for example, healthcare institutions, it should produce adequate results even in the average case without requiring large computational resources, a lot of time, or deep expertise.

There is novelty in modifying the work of Chen et al. [1] to suit tabular datasets as many state-of-the-art synthetic data generation models at the time of writing have been tested on image data, which in the context of real-world health data analysis is only a small portion. In addition, many of the image datasets used in literature are well studied and, being images, are autocorrelated and as such may not be realistic proxies for classification tasks in the real world as almost perfect classifiers can be

trained on them. The work of Chen et al. [1] uses some of the most widely studied benchmark datasets; the MNIST-handwritten numbers dataset [27] and the Fashion MNIST dataset containing black-and-white pictures of clothes [28].

From a more theoretical point of view, the question of whether the use of the privacy amplification by subsampling (PABS) technique [25] on tabular data provides similar benefits to privacy as with image data is important, because this technique plays an important role in making privacy guarantees of state-of-the-art strength obtainable with a GAN in the work of Chen et al. [1], while preserving high utility and sample quality. This technique, in the manner it is used in their implementation, has not been tested with tabular data. The way PABS is applied to GAN training in both the GS-WGAN [1] and this work is by training different parts of the GAN network with mutually exclusive subsets of data to induce more randomness, resulting in better privacy guarantees. This splitting of data could well have very different effects for tabular data in comparison to image data. For example, the question of how many subset divisions and consequently how much privacy benefit can be achieved with tabular data is interesting, as unlike image data, tabular data consists of several different types of features that are not necessarily parts of a larger whole as the features of an image (pixel values) are.

To summarize, the main objective of this thesis is to modify the GS-WGAN method presented by Chen et al. [1] to generate differentially private synthetic tabular data in an EHR context with strong differential privacy guarantees. The research questions asked are:

1. Is it possible to use this method to create DP synthetic tabular data with strong privacy guarantees that would retain downstream modeling utility?

2. Is it possible to use this method to create DP synthetic tabular data with strong privacy guarantees that would retain realistic sample quality?

3. Can the privacy amplification by subsampling [25] technique be used to enable strong privacy guarantees for tabular data as has been shown [1] with images?

1.4 Thesis contents

The contents of this thesis are as follows: chapter 2, *Machine learning*, covers, in a general manner, the necessary background related to neural networks (NN), generative modeling and generative adversarial networks (GAN). Chapter 3, *Differential privacy*, introduces the theory of differential privacy (DP) and the specific formulations of DP used in this thesis. In chapter 4, *Materials and methods*, detailed explanations of the methods used relating to DP and the GAN implementation are given and the data used is presented. Chapter, 5, *Related works*, ties the research in this thesis to other works in the field of DP synthetic data generation with GANs. Chapter 6, *Overview of models and experiments* presents detailed descriptions of the experiments used to acquire the results presented and discussed in chapter 7, *Results*. Finally, a summary, together with implications for future work are given in the chapter 9, *Conclusion*.

2 Machine learning

Machine learning (ML) (see for example [29]), refers to techniques that learn from data based on some measure of performance. Given a task, this can be generally described as learning a function f based on the *training data* used to accomplish it. Importantly, however, the goal is not to find the best fitting f for the training data, but a f that *generalizes* over new, unseen data. A situation where a model performs well when tested with the training data, but generalizes poorly to new data is referred to as *overfitting*.

In the case of generative ML models, which are at the center of this work, the task is to learn a good estimate of the underlying or *latent* "true" data distribution p_{data} [23]. This p_{data} is what is in statistics would be called the population distribution; the distribution samples of data are sampled from. In reality p_{data} is always estimated by using training data as we only have access to some small sample of all possible examples belonging to it. This training data sample follows what is called an *empirical distribution*, denoted in this work as $\hat{p}_{data}$. If a reasonably close representation to p_{data}, denoted p_{model} is learned by the algorithm, new data points that follow p_{model} such as synthetic patient data can be sampled from the generative model.

A standard way to categorize machine learning methods is to divide them into supervised and unsupervised learning algorithms (see for example [29]). Supervised learning methods can be characterized as methods that attempt to learn a mapping

from inputs $\boldsymbol{x}$ that are features of the observations of the training data $\mathcal{X}$ of size n to outputs $\boldsymbol{y}$, that are labels associated with the observations. During training the algorithm is given pairs of training data: $\boldsymbol{x}, \boldsymbol{y} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots (\boldsymbol{x}_n, y_n)\}$. The algorithm tries to predict the label y associated with each $\boldsymbol{x}$ and a loss value $\mathcal{L}$ value is calculated based on the correctness of that prediction in comparison to the *ground truth*. This is to say, that in supervised learning, a function $f(x)$, that approximates some true unknown function $y = h(x)$ is learned given some performance measure called a *loss function*, denoted in this work as $\mathcal{L}$.

This loss value $\mathcal{L}$ is then used to update the model parameters to improve it. In the case of a classification task for example, the label of an observation informs what class the observation belongs to. This label, or ground truth, can also be some other value, like a real value, for example, in the case of a regression task. In the binary classification tasks of this thesis, and the definitions relating to them, the labels are expressed as 0 and 1. The loss functions in generative models represent a penalty for failing to learn an estimate p_{model} that approximates the distribution p_{data} based on training data following $\hat{p}_{data}$ [23].

Unsupervised learning aims to find patterns in the data without auxiliary information given in the form of, for instance, labels [29]. A typical unsupervised learning task is, for example, clustering, where samples are grouped based on some similarity measure. Depending on the task they are used for Generative Adversarial Networks (GAN), used in this thesis, fall in-between these two categories to a class called 'semi-supervised learning' [23]. In the context of this thesis, however, the training of a GAN network can be thought of a supervised learning task because all data used have labels and the difference between what is supervised learning and what semi-supervised is not relevant in terms of this work.

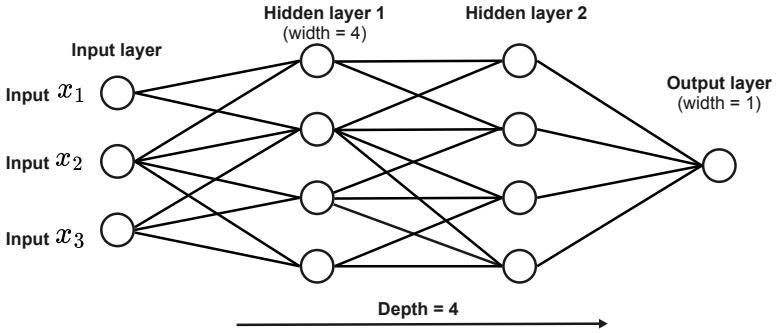

Figure 2.1: An illustration of a simple fully-connected neural network with related terms included

2.1 Neural networks

GAN models belong to a class of machine learning algorithms called *neural networks* (NN) and, more precisely, *deep neural networks*, which are NNs that have more than one hidden layer, that is, not an output or input layer [30]. The *architecture* of a neural network refers to it's structure. The *width* of a layer refers to how many nodes are in that layer and *depth* refers to the number of layers in the network. How the nodes are connected to each other is also an important choice in designing the architecture. The Figure 2.1 illustrates these terms and depicts a common simple type of neural network where the nodes are all connected to each other, called a *fully-connected* neural network. Connections can also be, for example, recursive, and some nodes might not even connect. (for more information see for ex. [30]).

The choice of architecture and the connections depend on the task at hand. Many complex architectures with different depths, widths, types of connections and units, that is, nodes that perform different calculations, exist. How the depth and width of the architecture affect NNs is not a straight-forward discussion; on the other hand, a simple network even with only one hidden layer, is a "universal approximator" [31] capable of, in theory, approximating any continuous function arbitrarily accurately, but networks with different depths can represent information at different levels of abstraction and may work better with some applications [32]. This capability of NNs

rises from using not only linear (dot products) but nonlinear transformations called *activation functions*. The format in which information is given to the model can also affect how deep or wide of a network is needed. One could, for example, compress the information of the data features using dimensionality reduction techniques or incorporate prior information about the data in the network by making specific architectural choices.

Training neural networks

The function defined by a neural network depends on learnable parameters $\boldsymbol{\theta}$, the most important being the weights $\mathbf{W}$ and biases $\mathbf{b}$, that are optimized based on the value of $\mathcal{L}$ for a given input. Learning in a neural network is achieved by finding values for $\boldsymbol{\theta}$ that minimize $\mathcal{L}$ through iterative updates to $\boldsymbol{\theta}$ on different inputs. The updates are guided by differentiating the loss function $\mathcal{L}$ on a given input with respect to the parameter values $\boldsymbol{\theta}$. In the context of neural networks and multiple input features these derivatives are gradients, denoted ∇, that is, vectors of partial derivatives. The mathematics surrounding NN's uses matrix notation and the gradients of a network are often expressed as *Jacobian matrices* [30]:

Definition 1 (Jacobian matrix)**.** *The Jacobian of the function* $f : \mathbb{R}^m \to \mathbb{R}^n$ *is the* $(n \times m)$*-dimensional matrix of elements* $\left(\frac{\partial f}{\partial x}\right)_{ij} = \frac{\partial f_i}{\partial x_j}$.

Feeding values through the network to get the loss given an input is called a *forward pass*, and it is the first phase of the *backpropagation* algorithm [33] used to calculate gradient information needed to update the network parameters. Generally, in one layer the following three things happen during a forward pass for one input $\boldsymbol{x}$:

1. The dot product, (generally $\mathbf{a}^\top \mathbf{b} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i$) between inputs $\mathbf{x}$ to a layer and the weights $\mathbf{W} \in \boldsymbol{\theta}$ of the layer is calculated:

$$\mathbf{W}^T \mathbf{x}$$

2. A bias term $\mathbf{b} \in \boldsymbol{\theta}$ is added to the result. The result of the dot product and addition of the bias is often denoted with $\mathbf{z}$:

$$\mathbf{z} = \mathbf{W}^T\mathbf{x} + \mathbf{b}$$

3. A nonlinear function called an activation function, often denoted with g is applied to the previous result: $g(\mathbf{z})$ and the result of this computation is then given as input to the next layer.

Often, and in the implementations of this thesis, the iterative updates to $\boldsymbol{\theta}$ are done using some version of Gradient Descent (see for example [30]). In the version used in this work, the inputs are given in batches, a collection of m input-output pairs sampled randomly from the training dataset. To be specific, in the version called Stochastic Gradient Descent (SGD), the parameter update is done for one input pair $\boldsymbol{x}, y$ at a time, and the version of the algorithm where this is done over a batch is called Mini Batch Stochastic Gradient Descent, but in practice, and in this thesis, both are still referred to as stochastic gradient descent as is the case in, for example, [34]. An *epoch* refers to iterating once over all $\lfloor n/m \rfloor$ of these batches constructed from the training data. The general form of a Gradient Descent update, with a *learning rate* α that determines the size of the update done to the parameters of the network for one iteration of SGD can be expressed as:

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{\text{old}})$$

The second phase of the backpropagation algorithm, called a backward pass [33], supplies SGD with the gradient information needed for the update. The calculation of the gradients of the loss averaged over a given batch with respect to the parameters of the network can be expressed as:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right)$$

Backpropagation is based on recursively applying the chain rule of derivatives: $\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}}{\partial y} \cdot \frac{\partial y}{\partial \mathbf{x}}$. In the context of machine learning the parts of the rule are often referred respectively to as downstream gradient = upstream gradient × local gradient. It can be helpful for intuitively understanding backpropagation and its connection to the chain rule of derivatives to express a NN as a composition of functions, chained together. For example, a NN with a depth of 3 can also be expressed as: $g(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$.

Definition 2 (The ReLU activation function)**.**

$$ReLU(z) = max(z, 0)$$

Figure 2.2 illustrates backpropagation in a NN with a computational graph using as activations the ReLU activation function [35] (see Definition 2). The **s** in the figure refers to the "score" and depicts the information propagated back to the layer in question from other layers closer to the output. This highlights how the process generalizes straightforwardly to networks of any depth. In other words, there could be any number of layers, and the calculation would not change whether the upstream gradient information coming from the next layer would be calculated based on the $\mathcal{L}$ or the "score".

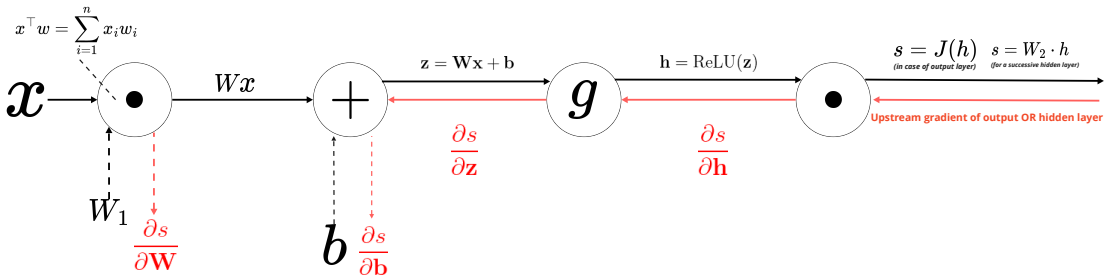

Figure 2.2: A high-level depiction of backpropagation as a computational graph. Black arrows depict the forward pass whereas red ones depict the backward pass.

The choice of a loss function is crucial for successfully training a NN and depends on the task at hand. As an example of a commonly used loss function for binary classification, and in GAN models, is the binary cross-entropy loss-function (A.K.A log loss) [36] shown in Definition 3.

Definition 3. *The average binary cross-entropy loss for m examples in a batch is:*

$$loss(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h\left(\mathbf{x}^{(i)},\boldsymbol{\theta}\right) + \left(1-y^{(i)}\right)\log\left(1-h\left(\mathbf{x}^{(i)},\boldsymbol{\theta}\right)\right)\right]$$

where $\log h\left(\mathbf{x}^{(i)},\boldsymbol{\theta}\right)$ *is the log-probability of the label assigned to an input being 1 given the state of the parameters* $\boldsymbol{\theta}$*. The formula becomes more intuitive if read similarly to an if-then statement: when the ground truth* $y^{(i)}$ *is 1 everything to the right of the addition goes to 0 and vice versa when* $y^{(i)}$ *is 0.*

ReLU [35] (see Definition 2) is a commonly used [30] activation function in GAN and other modern NN's. There are many different activation functions, and the choice of an activation function depends on the task. However, all share some properties: they must be differentiable and should be non-linear transformations. What is also important is that activation functions be such that they do not cause the values in the network to vanish or explode, that is, to start diminishing towards 0 or growing to infinity when values are multiplied in succession during a forward pass. There exist slightly different versions of the ReLU activation function, one of which is the *leaky ReLU*, which has a non-zero slope when $z < 0$. The additive slope term α is usually set to be a very small value, like 0.01 [37]. Some versions like the parametric ReLU (PReLU) also make this α a learnable parameter.

Another activation function commonly used in GAN networks is the hyperbolic tangent (Tanh): $g(z) = Tanh(z)$. Unlike the ReLU the values Tanh returns can be negative with a range between $(-1, 1)$. Whether a layer is a hidden, input or output layer, along with many other factors, affect the choice of the activation function. For example, some activation functions, like the *softmax* that scales values to be

between 0 and 1 (see for ex. [30]), are used only in the output layers.

2.2 Generative models and GAN

Generative Adversarial Networks [36] are a type of generative model where training is formulated as a competitive game between two networks: a generator G and a discriminator D. The G can be used to generate samples from the distribution it has learned, that is, p_{model}, by feeding a noise vector z to the network as input: $G(\mathbf{z})$. D discriminates between this generated output $G(\mathbf{z})$ and real data. In the context of GAN models that use as a loss function the Wasserstein-loss [26], such as the GS-WGAN [1], the discriminator is called a critic, but in this thesis it is referred to as a discriminator to avoid confusion and introducing extra terminology.

A typical example given to intuitively explain the relationship between G and D is the allegory of G being a forger (for example, forging money) and D being police that compares the outputs (forged money) of G to actual data (real money), attempting to separate these from each other. Albeit easy to understand, this may give a false impression, as, in the GAN training process, the discriminator shares information about its ability to make the distinction with the generator, which the G uses to improve its outputs updating its parameters with SGD. The problem is, that no police would tell a forger how they can be fooled more effectively and a comparison of a student and a teacher would perhaps be more accurate.

The popularity of the GAN approach is due to these models being able to produce good quality samples when compared to other generative models [23]. Many generative models can and are formulated as *Maximum Likelihood estimators* (MLE). *Likelihood* is the probability a model assigns to the training data given the parameters of the model $\boldsymbol{\theta}$. For a dataset of n examples MLE can be expressed as in Definition 4:

Definition 4. *Maximum Likelihood estimation*

$$\arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p_{model}\ \left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

where p_{model} is the probability distribution that the model assigns to $\boldsymbol{x}^{(i)}$.

As said, the goal of GANs is to implicitly learn an estimate of the data distribution through learning appropriate weights [23]. The cost functions of GANs are often formulated as *divergences* that quantify the proximity between the probability distribution estimated by the model p_{model} and the empirical, training set approximation $\hat{p}_{data}$ of the true population distribution. This use of divergences as loss functions works through formulating MLE as minimizing a divergence. For example minimizing the Kullback-Leibner (KL) divergence, that many GAN losses, especially older ones, are based on can be formulated as a MLE estimation as follows: $D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{\mathrm{x}\sim P}[\log P(x) - \log Q(x)]$, and is equivalent [23] to minimizing the earlier log-likelihood formulation (see Definition 4) of MLE:

$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}\left(p_{\mathrm{data}}(\boldsymbol{x}) \| p_{\mathrm{model}}(\boldsymbol{x}; \boldsymbol{\theta})\right)$$

Thinking about generative models in terms of maximum likelihood allows for making a distinction between implicit and explicit density models along the lines of the taxonomy of generative models presented by Goodfellow [23]. Explicit density models, for example Variational Autoencoders [38], or Deep Belief nets [39] estimate the distribution by choosing some model and using MLE to explicitly estimate $\hat{p}_{data}$. The challenge with the explicit approach is the difficulty associated with finding a model complex enough to express $\hat{p}_{data}$ while retaining computational tractability in fitting the model, especially when estimating high-dimensional distributions [23].

GAN models fall into the second category; implicit density models. The advantage that comes with this is, that they do not need to explicitly define a density function [23] as G only interacts with $\hat{p}_{data}$ indirectly by receiving information from

D. Not only is this beneficial in terms of avoiding the pitfalls of explicit estimation, it can be an advantage in terms of privacy, and is also thought to be one of the key properties that enable GANs to work as well as they do. It has been argued that this property also could make GAN models more resistant to overfitting than explicit models [23].

Training GANs and game theory

Just as there are no free lunches, the implicit approach does not come without drawbacks. Firstly, although theoretically G can be any differentiable function with learnable parameters and does not have to be a neural network [23], in practice, NNs are the approach used. Like all deep NNs, they are a "black box" in that their inner workings are hard to interpret. There is also a lot of variance between which minima of the loss function space different training runs may converge to. Second, as the optimization process (via SGD) depends on both the parameters of the generator $\boldsymbol{\theta}^G$ and the discriminator $\boldsymbol{\theta}^D$, which both change during the process, the cost function is not stable. In other words, and in comparison to, for example, a classifier with a cost-function like BCE (see Definition 3), the "goal posts are moved" during training G, because the loss function depends on the parameters of D, which are also updated. This is part of why GAN models, especially older ones using the BCE-loss, can be hard to get to converge [26]. If, however, training proceeds in a balanced manner, a good estimate of the ratio between the two distributions $\frac{\hat{p}_{data}(\mathbf{x})}{p_{model}(\mathbf{x})}$, can be reached, with which the network can be trained. This dynamic of change and instability is what sets adversarial learning and GAN models apart from other generative models.

From the changes to both network's parameters during training and the formulation of GAN as a game it follows, that the optimum of a GAN is not the optimum of a cost function, but a *Nash equilibrium* [23] of a *minimax-game* between G and

D [36]. This is what is "adversarial" about adversarial learning. Informally and for the sake of intuition, one can think of a Nash equilibrium in a two-player game to be a situation where both players know each other's strategies, but it is not beneficial for either to change strategy. In other words, a player can not improve their chances of winning by changing just his/her strategy. A classic example would be the prisoner's dilemma (for more information see [40]).

Formulating GAN training as a minimax game firmly links GANs to game theory. Adopting the terminology of Nash equilibria in continuous games used by Ratliff et al. [41], we can summarize the GAN training process with some arbitrary, but separate loss function $\mathcal{L}$ for the two network's as follows:

- The discriminator's goal is to minimize $\mathcal{L}^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)})$, while only having control over it's own parameters $\boldsymbol{\theta}^{(D)}$.
- The generator's goal is the opposite; to minimize $\mathcal{L}^{(G)}(\boldsymbol{\theta}^{(G)}, \boldsymbol{\theta}^{(D)})$, while only having control over $\boldsymbol{\theta}^{(G)}$.
- The Nash equilibrium is a tuple of states of the parameters $(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)})$, that is a local minimum of $\boldsymbol{\mathcal{L}}^{(D)}$ with respect to $\boldsymbol{\theta}^{(D)}$ and a local minimum of $\boldsymbol{\mathcal{L}}^{(G)}$ with respect to $\boldsymbol{\theta}^{(G)}$

Common cost functions can be modified to fit the minimax game setup. The BCE-loss, for example, (see Definition 3), in the context of a game can be written as:

$$\min_D \max_G -[\mathbb{E}(\log(D(x))) + \mathbb{E}(1 - \log(D(G(z))))]$$

Challenges in training GANs

The instability of GAN training discussed before often manifests itself during training as oscillating losses of the two networks D and G. This can make the loss hard

to interpret, and the inability to balance the learning of D and G during training can cause different failure states. Often training D is simply easier. The reason for this is the difference in the difficulty of the two networks' tasks. G has to produce complex output, whereas D is often a relatively simple classifier model. One can try to intuit this difference through the forgery example seen previously: it is much easier to distinguish fake from real money than to forge money successfully. In general, the D network needs to be better at its task than G in order to supply it with meaningful information for updates. For this reason, the D is often trained multiple times per one training step of G.

Perhaps the most problematic prominent failure-state is *mode collapse*. Mode collapse happens when the generator begins to only produce outputs from some mode of the distribution. This happens if fooling the discriminator with these outputs is the easiest way to lower the loss. As an example, consider the MNIST dataset [27], consisting of grayscale pictures of handwritten numbers 1-9. To simplify, consider a situation where we are training a generator to learn to produce pictures similar to the pictures of digits only from classes 1, 7, and 3. This setup would be prone to mode collapse since the 1 and 7 are close to each other compared to 3. If G gets good results by generating only pictures of 1 or 7, which D has a hard time distinguishing, the training process can get stuck in a local minimum. This is also an example of how the update of one network can direct the updates of the other one in adversarial training.

Having covered the prerequisites concerning neural networks and GANs, we are now ready to move on to the theory surrounding Differential Privacy (DP), building toward an understanding of how to train a GAN in a Differentially Private manner.

3 Differential privacy

The first definition of Differential Privacy, known today as 'pure' or 'ϵ, 0' Differential Privacy was proposed by Dwork and Mcsherry [16] in the early 2000's. The development leading to this definition was, however, the result of a long line of work, influenced especially by for example the ideas of Dinur and Nissim (see for example [42]). The literature surrounding DP is extensive, and many different formulations of DP are available, coupled with research on its application to different problems. This thesis adopts a formulation of DP called Rényi Differential Privacy (RDP) [43]. As we will see later, Rényi DP is a 'relaxation' of the 'pure' definition, which offers benefits in the context of DP-training of neural networks.

This chapter is arranged so, that it builds towards the definition of RDP [43] by comparing it to the older, but closely related and widely used form called ϵ, δ-DP [44] shown in Definition 6. Conveniently, ϵ, δ-DP also encompasses the original definition in the case where $\delta = 0$, which is also where the name $\epsilon, 0$-DP stems from. Before further discussing the intuition behind DP, let us see the definition for what is an adjacent dataset (see Definition 5) as well as the definition of ϵ, δ-DP (see definition 6 mentioned above to lay a mathematical foundation for explaining DP in terms of this work.

Definition 5 (Adjacent sets of Data)**.** *Let D be a set of data consisting of observations, that is, vectors of feature values $\{d_1, d_2, \ldots d_n\}$. A set of data D', consisting of observations $\{d'_n, d'_n, \ldots d'_{n-1}\}$ that differs from D by one observation is said to be*

adjacent to D.

Definition 6 (ϵ, δ- Differential Privacy)**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow$ with range $\mathcal{R}$ satisfies (ϵ, δ) - differential privacy if for any adjacent sets of data D and D' that differ by one observation given as input, and any subset of outputs $S \subseteq \mathcal{R}$ it holds that*

$$Pr[\mathcal{M}(D) \in S] \leq \in e^{\varepsilon} Pr[\mathcal{M}(D') \in S] + \delta$$

Where $\mathcal{M}$ is in, for example, this thesis, the GAN training algorithm, ϵ is the upper bound for privacy loss, and δ is the probability of breaching the DP-guarantee.

3.1 The differential privacy guarantee

As the name of the theory hints, privacy in DP is understood as something "differential," e.g., relating to a difference. What this means in the context of an algorithm can be illustrated by considering, for example, the calculation of a mean from some randomly chosen subset of data, which in this example would be the randomized mechanism $\mathcal{M}$ (see Definition 6). It is perhaps good to highlight, that the reason the algorithm must be randomized, and not deterministic is, because in the case of a deterministic algorithm, it would always be known which observations take part in the calculation and adding perturbation would be futile. For example, in the linkage attacks described in the introduction, group membership could always be known for certain if an adversary was holding auxiliary information.

The output of the mechanism $s \in S$ depends on the mechanisms input data D and if we apply this mean mechanism to adjacent datasets D and D' that differ by one element (see Definition 5), and there is some difference in the probability of some result $s \in S$, this operation can be seen to have revealed information about

the data used. This difference in probability of the outputs of $\mathcal{M}$ is, how privacy is formalized as a mathematical definition in DP. As put by Chen et al. [1] Differential Privacy "guarantees the difficulty of inferring the presence of an individual in the private dataset by observing the outputs of the mechanism $\mathcal{M}(D)$.

To give intuition to what this means, Differential Privacy can be, in the context of data that concerns human subjects, said to "protect the sensitive data of individuals from additional harm that might come from their data being used in a computation versus it not being used" [10]. From the point of view of an adversary, the amount of information that can be learned about any individual's data is now bounded by the privacy loss, denoted with (ϵ), whether that individuals data was part of the private data used in the calculation or not. The aim of DP is to protect against learning too much about any individual, but not against learning general patterns from the data. In other words, and informally, it can be said that the DP guarantee is a upper bound for the amount of information that can be learned only from one individual's data and can not be inferred from other observations.

To bridge the gap between this intuition and the ϵ, δ-DP formulation we can rearrange Definition 6, to illustrate how ϵ bounds the difference of probability of outputs:

$$\log \frac{Pr(\mathcal{M}(D) \in S)}{Pr(\mathcal{M}(D') \in S)} \leq \epsilon + \delta$$

The parameter ϵ is the upper bound we give to the maximum privacy loss allowed for the guarantee to hold over all possible D and D''. When the guarantee holds, the output of the mechanism is said to be ϵ, δ differentially private for the values of $epsilon, \delta$. Informally and for the sake of intuition, one can think that when calculating ϵ, we consider all possible D and D', meaning that each observation is, in its turn, the differing record between the "possible" adjacent datasets that can be crafted. This is not how the computation is done in practice, as we will see later, but

the simplified idea may help with grasping, why DP guarantees have properties that also concern auxiliary information and future developments in attacks, as discussed in Section 1.2.

A higher value of ϵ allows for more privacy loss. Leaning on Definition 6 we can see that an epsilon value of 0 would mean that for every input, the output would be the same, that is to say, independent of the input. The δ parameter accounts for some rare event where the guarantee would be broken, for example, by chance. Incorporating this small "probability for catastrophe" in the definition allows for greater flexibility in choosing how to apply DP and can be used to achieve closer bounds for privacy loss [43]. δ should be a very small value, according to Dwork [17] "... smaller than the inverse of any polynomial in the size of the database".

Calibrating noise to sensitivity

One of the key innovations of the work of Dwork and McSherry [16] was, how calibrating the amount of perturbation added to the mechanism can be linked to a value of ϵ. How this perturbation is done can be many things, but often, it is noise sampled from some distribution and applied via a *noise mechanism*. In this thesis's models, for example, the Gaussian noise mechanism is used. With this noise mechanism, the noise is sampled from a Gaussian distribution; $\mathcal{N}(0, \sigma^2)$. In comparison to many SDC methods that perturb, but don't measure the effects of perturbation, calibrating perturbation to a bound gives a way to control the privacy risks and also makes it possible to know how much noise one needs to add to achieve guarantees of different strengths.

In the literature, what is referred to when speaking about a *mechanism* $\mathcal{M}$ is not always clear. In the context of statistical databases and for example, the work of Dwork [17] $\mathcal{M}$ is often used to refer to only the perturbation process, for example, the application of Gaussian noise to satisfy DP [17]. However, in the context of

differentially private machine learning literature, what is the 'noise mechanism' and what refers to the whole computation of the algorithm is often not the same as with statistical databases. The word "mechanism" in DP training of ML models is usually used to refer to the whole computation, for example, by referring to a 'training mechanism' in the context of training models via a differentially private form of SGD called differentially private stochastic gradient descent [34] (DPSGD) used in this thesis. In this work, a distinction is made between referring to the mechanism $\mathcal{M}$ as the whole DP-computation and the perturbation method used in this training mechanism as the "noise mechanism". A *run* of the mechanism refers to a single run of the mechanism, for example, one iteration of optimization via DPSGD over one batch and not the whole fitting process of the model.

To calibrate the noise to some value of ϵ, the parameters of the noise mechanism are tuned according to the *sensitivity* of the function DP is applied on. Sensitivity can be defined in many ways, but is essentially a distance, specifically the maximum distance of how much output can change. In terms of the Gaussian mechanism the ℓ_2-sensitivity, denoted as Δ_2 [17], based on the euclidean distance is used. Increasing the value of *epsilon* will allow the maximum distance between outputs to be greater.

Definition 7 (ℓ_2-sensitivity and the Gaussian Mechanism [17])**.** *Let* $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^d$ *be an arbitrary d-dimensional function, and define its* ℓ_2 *sensitivity to be* $\Delta_2 = \max_{adjacent\ x,y} \|f(x) - f(y)\|_2$*. The Gaussian Mechanism with parameter* σ *adds noise scaled to* $\mathcal{N}(0, \sigma^2)$ *to each of the d components of the output, where* σ *depends on the sensitivity* ℓ_2 *based on the formulation of DP used, for example, in the case of Rényi DP see* 12*.*

Other sensitivity measures and noise mechanisms are, for example the Laplace mechanism [17], paired with the ℓ_1-distance (taxicab) metric. The choice of how sensitivity is bounded depends on the task; in the case of multi-dimensional vectors (such as gradients), the $l2$-sensitivity metric, can be used to achieve lower values for

sensitivity than the $l1$-distance, whereas for counting queries (for ex. counting the number of observations in a group) the $l1$-distance with laplacian noise is often used as the distance between queries is one-dimensional [17].

Composition of mechanisms

Many tasks, like the DP-optimization of a NN using DPSGD [34] require many sequential runs of a mechanism. To keep track of the total privacy loss over these sequential applications, different *composition theorems* have been formulated. The total amount of privacy loss is referred to as the *privacy budget* [17]. In the case of ϵ, δ-DP, a simple way to keep track of this budget is to use the basic composition theorem (see Definition 8) that allows for straightforward addition of ϵ values of sequential mechanisms [17]. In the GAN implementation of this work, however, *advanced composition theorems* are used. The details of these theorems are out of this thesis's scope, and an understanding of them as improvements to the basic composition shown in Definition 8 to get tighter bounds of ϵ and smaller privacy budgets is sufficient.

Definition 8 (Basic composition for (ϵ, δ) - DP [17])**.** *Let $\mathcal{M}_i : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}_i$ be an $(\varepsilon_i, \delta_i)$-differentially private algorithm for $i \in k$, where i is the amount of runs of the mechanism. Then if $\mathcal{M}_k : \mathbb{N}^{|\mathcal{X}|} \to \prod_{i=1}^{k} \mathcal{R}_i$ is defined to be $\mathcal{M}_k(x) = (\mathcal{M}_1(x), \ldots, \mathcal{M}_k(x))$, then the total of all k runs is $\mathcal{M}_k$ is $\left(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i\right)$ differentially private.*

3.2 Rényi differential privacy

Equipped with the prerequisite knowledge, let us introduce the formulation of DP used in the models of this thesis: Rényi Differential Privacy (RDP) introduced by Mironov [43]. RDP is a relaxation of the ϵ, δ formulation and is based on the Rényi

Divergence (see [45]):

Definition 9 (Rényi Differential Privacy (RDP) [43])**.** *A randomized mechanism $\mathcal{M}$ is (λ, ϵ) - Rényi differentially private with order λ, if*

$$D_\lambda(\mathcal{M}(D)||(D')) = \frac{1}{\lambda - 1} log\, \mathbb{E}_{x\sim\mathcal{M}(D)} \Big[\Big(\frac{Pr[\mathcal{M}(D) = x]}{Pr[\mathcal{M}(D') = x]}\Big)^{\lambda-1}\Big] \leq \epsilon$$

holds for any adjacent datasets D and D', where

$$D_\lambda(P\,||\,Q) = \frac{1}{\lambda - 1} \log\, \mathbb{E}_{x\sim Q}\big[Pr[\big(P(x)/Q(x)\big)^\lambda\big] \leq \epsilon$$

Different formulations of DP are not exclusive and relate to each other closely. For example the privacy budget of Rényi DP can be converted [43] to that of ϵ, δ-DP as follows:

Definition 10 (Connection between the upper bounds of $(\epsilon, \delta) - DP$ and RDP)**.** *A λ, ϵ-RDP mechanism $\mathcal{M}$ is also*

$$(\epsilon + \frac{log 1/\lambda}{\lambda - 1}, \delta) - DP$$

Although closely related, Rényi DP improves some aspects of ϵ, δ-DP for some tasks, such as GAN training, and allows for more advanced composition theorems to be used to achieve a closer bound for ϵ. Without delving too deep it can be said that one of the main reasons RDP is beneficial to use with the GAN implementation in this work instead of ϵ, δ-DP relates to a problem with large domains [43], such as the domain of the cost function of a NN. Informally, and reasoning based on the *curse of dimensionality* (see [46]), the amount of probability we have to allow under δ grows because the probability in the tails of the distribution of ϵ grows exponentially [43] with the growing amount of "space" going to higher dimensions. In terms of this

work, adopting RDP alleviates this problem by using a divergence based measure and allowing for the use of a more efficient RDP-*privacy accounting technique* [47], that tracks the higher moments of the distribution of ϵ during the DP training of the network. This, however, is a complex and lengthy subject, which is not required to understand the contents of this thesis. Interested readers can find more information on advanced privacy accounting via RDP, for example, in [47].

Applying DP to synthetic data generation

In terms of privacy, repeated queries to sensitive data are problematic. This is a consequence of the fundamental law of information recovery, described by Dwork [17] as follows: "overly accurate answers to too many questions will destroy privacy in a spectacular way". In terms of adversarial attacks, and this thesis, the consequences of this law [17] are, that the more optimization iterations are allowed, privacy protection erodes quickly.

Another important consideration in terms of privacy protection is what is released and what privacy guarantees are given over. As we will show in the methods section (see Chapter 4), in terms of the GAN network, the privacy guarantee is given over the generator network G. Due to the postprocessing property [17] (see Definition 11) of DP, these guarantees are also extended over the synthetic data generated by a differentially private G [1]. In more broad terms, the post-processing property guarantees that the results of all computations on a DP-guaranteed output are also DP. This is part of what gives rise to DP being resistant to any future attacks, as discussed in the introduction (see section 1.2), and makes generating differentially private synthetic data meaningful as any results derived from it will, generally, also be under the DP-guarantee [17], of the same strength as the guarantee over the G.

Definition 11 (Post-Processing [17])**.** . *Let $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R$ be a randomized algorithm that is (ε, δ)-differentially private. Let $f : R \to R'$ be an arbitrary randomized*

mapping. Then $f \circ \mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R'$ *is* (ε, δ) *differentially private.*

4 Materials and methods

This section covers the methods used to generate differentially private synthetic data in this work. The data and pre-processing steps taken are also described. The implementation of the GAN model in this thesis is a modification of the GS-WGAN [1] to suit tabular data generation. The implementation that was the basis of this work can be found in [48]. The changes made comprised of designing a new network architecture to suit tabular data generation and making the needed changes to how the gradient information used in the DP-optimization of the network is calculated and how perturbations were performed for this new architecture and tabular data. To summarize the differences in the implementation of this thesis and that of Chen et al. [48], it can be said, that while technical changes due to different data dimensionality were made and architectural choices were different, the way in which DP is applied and the network is trained follow the work and ideas in presented in [1].

4.1 Differentially private stochastic gradient descent

In this thesis, the GAN network is trained using differentially private stochastic gradient descent (DPSGD) first introduced by Abadi et al., [34]. DPSGD achieves DP guarantees through manipulating the gradients related to each training step during model training. First, the sensitivity Δ_2 of the run of the mechanism, that is, one step of training, is bounded by *clipping* the gradients to some maximum

magnitude determined by the *clipping bound* C. After clipping, calibrated noise is added to the gradients calculated with backpropagation for the batch of input data [34].

The bounding of sensitivity by clipping is what enables DP-guarantees to be calculated for the DPSGD-optimization of the GAN network. Similarly to Chen et al. [1] we will refer to the process of applying clipping and noise to the gradients as *sanitation*. In the definitions of this chapter, the subscript of the sensitivity notation Δ_2 is sometimes left out in order to avoid multiple subscripts and improve readability, but is still used to refer to the ℓ_2-distance based sensitivity.

A DPSGD training step, similarly to as presented in [1] can be summarized as follows:

1. Gradients before sanitation are calculated with backpropagation as in non-DP SGD. At training step t these are: $\nabla^t_{\mathcal{L}}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}_D, \boldsymbol{\theta}_G\right)$

2. Sanitation of gradients is done by clipping and adding noise:

$$\hat{\boldsymbol{\nabla}}^t = \mathcal{M}_{\sigma,C}\left(\boldsymbol{\nabla}^{(t)}\right) = \text{clip}\left(\boldsymbol{\nabla}^{(t)}, C\right) + \mathcal{N}\left(0, \sigma^2 C^2 \boldsymbol{I}\right)$$

3. The parameters of the model are updated using the sanitized gradients as in normal gradient descent: $\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} - \eta \cdot \hat{\boldsymbol{\nabla}}^{(t)}$

As noted previously in the Differential Privacy chapter, noise calibrated to the bounded sensitivity value comes (in our case) from a Gaussian noise mechanism. The result of using this noise mechanism with RDP for the training of the model [43] is described in the Definition 12 below.

Definition 12 (RDP using a Gaussian noise mechanism [43]). *Let $f : X \to \mathbb{R}^d$ be an arbitrary d-dimensional function with sensitivity being*

$$\Delta_2 f = \max_{S,S'} \|f(S) - f(S')\|_2$$

over all adjacent datasets D and D'. The Gaussian Mechanism $\mathcal{M}_\sigma$, parameterized by σ, adds noise into the process

$$\mathcal{M}_\sigma(x) = f(x) + \mathcal{N}\left(0, \sigma^2 I\right)$$

the output of this $\mathcal{M}$ is $\left(\lambda, \frac{\lambda \Delta_2 f^2}{2\sigma^2}\right)-$ RDP.

4.2 Wasserstein GAN and gradient penalty

In the implementation of this work, the Wasserstein loss function, abbreviated as Wloss, is used, together with a Gradient-penalty (GP) term [26]. In the literature, a Discriminator of a GAN-network that uses the WLoss is called a critic (see original paper by Arjovsky et al., [49]) because it outputs a continuous, real-valued score of distribution closeness and not a probability-like estimate as in the case of BCE-loss (see Definition 3). As mentioned in this thesis, however, D is still used to refer to the critic to avoid confusion and introducing extra terms.

Wloss approximates a linear function called the Earth-Mover's distance (for detailed information in relation to GANs see [49]) or Wasserstein-1 distance. The intuition behind the the EM-distance can be described with the example of moving mass, for example a pile of soil, from one place to another. If we let $\gamma(x, y)$ indicate how much mass must be transported from x to y in order to transform the distribution $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$, the EM distance is then the "cost" of the optimal plan to transport the mass. Also, intuitively, moving mass is a linear operation; for example, the size of a spade used to move the soil does not grow as work proceeds, and so doesn't the amount of mass moved between iterations. From this linearity, it

follows that for Wloss to be a valid approximation of the EM-distance, a 1-Lipschitz continuity condition must hold [49] (see Definition 13). The "1" in the condition means, in terms of our application of this metric, that the gradient's magnitudes in training must be between $[-1, 1]$. Below are the definitions for Lipschitz-continuity, followed by the definition for the EM-distance:

Definition 13 (Lipschitz-continuity)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is globally L-Lipschitz continuous if there exists an $L \geq 0$ such that*

$$\|f(x) - f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^n$$

Definition 14 (Earth-Mover (EM) distance as in [49])**.**

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

where $\Pi\left(\mathbb{P}_r, \mathbb{P}_g\right)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$.

The Lipschitz-1 continuity condition is enforced with a gradient-penalty regularization term (GP) introduced by Gulrarajani et al. [26]. In terms of this thesis it can be seen as a "soft" regularization method, that pushes the magnitude of the gradient's towards $[-1, 1]$ when used with the loss function of D (see Definition 15).

The intuition behind the gradient penalty term is to sample by interpolating between the real $\mathbf{x} \sim \hat{p}_{data}$ and fake, $\hat{\mathbf{x}} \sim p_{model}$ samples, regulating the interpolation with the random parameter $\alpha \sim \mathcal{U}[0, 1]$ and calculating the gradient of the loss of the interpolated sample with respect to $\boldsymbol{\theta}_D$, penalizing gradient values with magnitudes far from $\mid 1 \mid$. Informally one can intuit this as feeding "approximated possible inputs" to D that consider both the state of the parameters of D as well as G. If the gradients of these "possible inputs" coming from different sources, are close to

1, then the gradients of the loss with respect to $\boldsymbol{\theta}_D$ as well as with respect to $\boldsymbol{\theta}_G$ will, as Chen et al. express it [1] be close to 1 "almost everywhere".

What "almost everywhere" means in terms of the models of this thesis is that the enforcement of the Lipschitz-continuity condition with GP [26] is not absolute, and the approximation of the EM-distance may be slightly flawed. However, this small error is allowed because the technique works in practice, and even if the gradient values would not be close to | 1 |, DP guarantees would still stand because of the clipping of the gradients. In other words, the negative effect due to inaccuracy would come to the convergence of the model to suitable parameters, as the proportion of the gradient information lost due to clipping would be greater, but would not affect the legitimacy of the DP guarantees. The Wasserstein-1 loss function with GP, used to train the GAN-network is expressed below in Definition 15, with loss calculation for G and D separated as presented in [1]:

Definition 15 (Wasserstein-1 loss with Gradient Penalty [26])**.**

$$\mathcal{L}_D = -\mathbb{E}_{\boldsymbol{x}\sim\hat{p}_{data}}[D(\boldsymbol{x})] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p_{model}}[D(\tilde{\boldsymbol{x}})] + \underbrace{\lambda\mathbb{E}\left[(\|\nabla_D(\alpha\boldsymbol{x} + (1-\alpha)\tilde{\boldsymbol{x}})\|_2 - 1)^2\right]}_{\textit{Gradient penalty term}}$$

$$\mathcal{L}_G = -\mathbb{E}_{\boldsymbol{z}\sim P_{\boldsymbol{z}}}[D(G(\boldsymbol{z}))]$$

Where $\alpha \sim \mathcal{U}[0,1]$ is the interpolation coefficient, $P_{\boldsymbol{z}}$ is the noise sampled from a normal distribution given as input to G to generate samples, λ is the regularization strength hyperparameter of the gradient-penalty term, $\boldsymbol{x}$ is the real data and $\hat{\boldsymbol{x}}$ is the data generated by G.

The Wloss offers benefits to training GANs compared to other loss functions, such as, the BCE-loss (see Definition 3) used with older gan models. First, the Wloss is more stable, and convergence is often faster [49]. Second, there are theoretical synergies, covered in Section 4.3, between the DPSGD algorithm and Wloss [1].

The benefits of stability and convergence can be perhaps, best explained from the point-of-view of the problems relating to BCE that the Wloss avoids. Most

importantly, the BCE loss function is more prone to vanishing gradient values and mode collapse than Wloss. This behavior is a consequence of the sigmoidal shape of the BCE and the unequal demands of the training process of D and G, that is, D's less complex task (binary classification) in comparison to the generation task of G. With BCE-Loss, the difference between predictions and the target increase linearly, but the loss can increase exponentially. In terms of a divergence between distributions this can be thought of as the distribution p_{model} and the empirical distribution that D approximates $\hat{p}_{data}$ causing large loss values when they are far from each other. This can result in training getting stuck at the plateaus at either end of the sigmoid shaped function, consequently causing the training to go into failure modes like mode collapse. The Wloss improves on the problem of vanishing gradients through its approximation of the EM distance measure [49]. As discussed, EM loss is linear and, as such, does not have the plateaus of the BCE loss; this means that the distance between p_{model} and $\hat{p}_{data}$ grows linearly with the loss values, resulting in training being more stable.

4.3 Privacy benefits of the approach

In addition to the benefits of using Wloss relating to training stability and convergence, there are theoretical synergies between the DPSGD training algorithm and the Wloss-function. As said, if the Lipschitz-1 condition holds, the sensitivity is "almost always", $\Delta_2 \leq 2$ (maximum distance between 1 and -1), which allows us to choose the clipping bound $C = 1$ for the DPSGD training algorithm. This results in less information loss due to clipping than other methods, where the gradient magnitudes are not known. The discovery of this synergy is one of the key innovations of Chen et al., [1] and makes the WLoss-function a good choice for training GAN models with DP.

Also other benefits follow from this discovery: first, applying DP and calculating

ϵ becomes easier since Δ_2 is bounded and fixed for every run of the mechanism. This gives us a data-independent privacy cost where each generator update step satisfies $(\lambda, 2b\lambda/\sigma^2)$-RDP where b is the batch size [1]. Second, there is no need for a hyperparameter search to find the optimal value of C, which can be a costly and difficult task in the case of unbounded gradients. These two properties are also beneficial in terms of usability as will be discussed later.

In addition to these advantages brought by architecture choices, GAN models in general also have another connection with privacy that rises from their game-like setup; the generator used to create the synthetic data does not see the data directly. This can be thought to lead to more resistance against generating (exact) duplicates of real data points as a result of overfitting.

In the work of Chen et al. [1], and in this thesis, this property of G having access to the sensitive data it is trained with only through D, is taken advantage of by selective application of the gradient clipping and noise addition to only the information that G receives from D. This "moving" of the clipping and noise addition from the side of D to G preserves more gradient information in comparison to multiple applications of perturbation or applying perturbation to the training of D as has been done in other recent work (see for example [50]). When the sanitation is done in this way, no perturbation is needed when training D, since the DP-guarantee is given over G and consequently its output, the synthetic data that is what we want to release. This setup also allows for pre-training discriminators without privacy-cost using a temporary generator, discarded after pre-training before training the private G.

This manner of sanitation can be thought of as placing a "privacy barrier" as put by Chen et al. [1] between D which sees the private data and G by perturbing $\nabla_G^{upstream}$ during backpropagation, but leaving other gradients, for example the local gradient of G and the gradients flowing from G to D unperturbed. Figures 4.1 and

4.2 illustrate the flow of information during training of both networks, giving an overview of the training process. Both pictures are inspired by the figures describing the sanitation process in the work of Chen et al. [1], but contain more information to try to illustrate also, on an abstract level the relationship between G and D. The notation $G(z, \boldsymbol{\theta})$ refers to a noise vector z being fed through the G network, which is how new samples are generated with GANs.

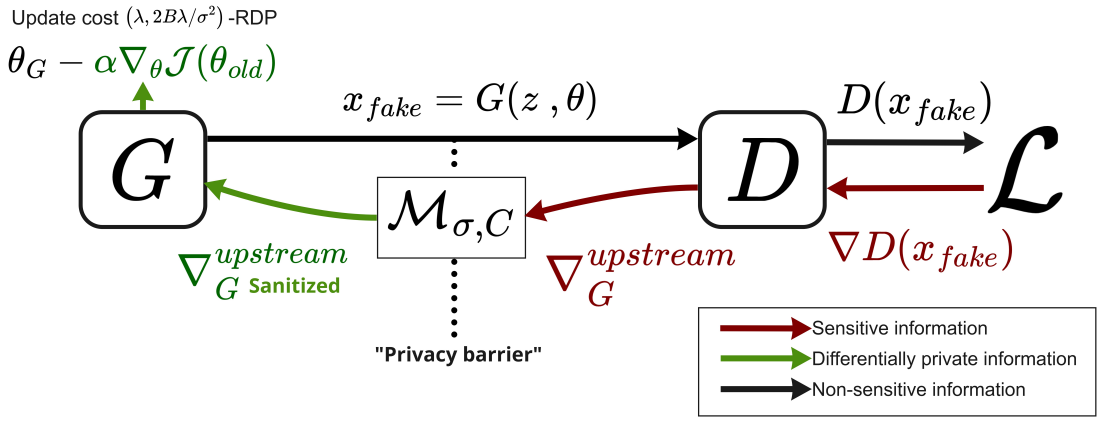

Figure 4.1: Flow of information for one DPSGD update step of the Generator. The information that flows from G to D is not sanitized, but the information from D, that sees the private data to G is. The notation $D(\boldsymbol{x}_{fake})$ refers in this picture to the "fake" examples produced by G being given to D to get the value for the loss. The information that is sensitive is marked with a red color, whereas non-sensitive information is marked with a black color. Green denotes sanitized information, that is information under a DP guarantee.

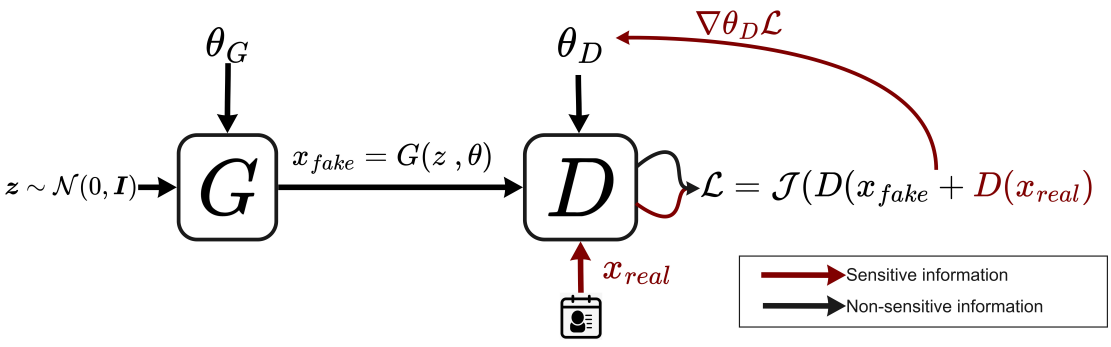

Figure 4.2: Flow of information during an update step of one D network. The information that is sensitive is marked with a red color, non-sensitive information is marked with a black color. D gets as input both "fake" and "real" examples and updates itself based on the loss that depends on it's ability to distinguish the origin of these inputs.

Privacy amplification by subsampling

Last but not least, there is one more privacy-related topic to cover used in this thesis and the work of Chen et al [1] to reach better privacy guarantees; the privacy amplification by subsampling (PABS) technique [25]. While amplifying privacy by subsampling the private data is a well-known and common procedure and is not specific to the implementation of GS-WGAN, Chen et al. [1] apply PABS to private GAN training in a creative way that takes advantage of the training of the D being free of privacy costs, as discussed before. To better understand what is, and what is not PABS, let us first see how it is already in effect due to dividing data into batches in DPSGD. The privacy cost of a DPSGD training step of G that is $(\lambda, 2b\lambda/\sigma^2)$ depends on the batch size b. The privacy benefit due to subsampling comes from the chance of leaking information about a particular individual now depending on the probability $\frac{b}{n}$, that is, whether that data point was involved in the step.

The way PABS is incorporated into training the GAN in the models used in this thesis is by using multiple discriminators, that are pre-trained independently of each other with mutually exclusive subsets of the data. During pre-training, we use a non-private, temporary G that is discarded after completing this phase. During the main training process, for each update of the private G (see Figure 4.1) one of the pre-trained D networks is chosen randomly to be queried in that update step. For subsequent update steps, a new D is again randomly chosen. This results in stronger privacy guarantees similarly to as was seen with training in batches, because now, for example, an adversary can not know for certain, which D is queried and what data points that D has previously seen.

The dynamic between the number of pre-training iterations and the primary DP-training process affects the stability of GAN training. Too many pre-training iterations can cause the G network to have difficulties learning. At the same time, too few raise the number of iterations needed to train G if the D networks can not

give meaningful information to G at the beginning of training as their parameters are still too close to the initialization values.

Now that the techniques used are covered, we will see how the total amount of Privacy Loss is calculated for the whole process depicted.

Accounting for privacy loss

How ϵ, that is the upper bound of the privacy loss of a $\mathcal{M}$ is calculated matters, in that the resulting ϵ might be looser than the "true" tightest bound for some $\mathcal{M}$. In this thesis, the privacy cost of the training process is tracked using the analytical moments accountant for subsampled RDP-mechanisms presented by Wang et al. [47] using the autodp-package (version 0.2) [51]. This approach uses advanced composition of RDP-mechanisms and tracks the privacy-loss distribution to give tight bounds for RDP guarantees [47]. The composition theorem for subsampled RDP mechanisms [47] is as follows:

Definition 16 (Composition of RDP for Subsampled Mechanisms [47])**.** *$\mathcal{M}$ is a randomized algorithm taking as input non-subsampled data and $\mathcal{M}\circ$ subsample is a randomized algorithm taking as input data subsampled without replacement with a subsampling rate $\gamma = m/n$*

For all integers $\lambda \geq 2$, if $\mathcal{M}$ is $(\lambda, \epsilon(\lambda))$-RDP, then $\mathcal{M}\circ$ subsample is $(\lambda, \epsilon'(\lambda))$-RDP and consequently in terms of $(\epsilon, \delta - DP)$ as follows: $(\epsilon + \frac{log 1/\lambda}{\lambda - 1}, \epsilon'\lambda)$

$$\epsilon'(\lambda) \leq \frac{1}{\lambda - 1} \log \left(1 + \gamma^2 \binom{\lambda}{2} \min \left\{ 4 \left(e^{\epsilon(2)} - 1 \right), e^{\epsilon(2)} \min \left\{ 2, \left(e^{\epsilon(\infty)} - 1 \right)^2 \right\} \right\} + \sum_{j=3}^{\lambda} \gamma^j \binom{\lambda}{j} e^{(j-1)\epsilon(j)} \min \left\{ 2, \left(e^{\epsilon(\infty)} - 1 \right)^j \right\} \right)$$

Advanced composition theorems are a whole line of research of its own and to discuss them on a deeper level a significant amount of prerequisite information would

have to be covered. Understanding these techniques is not crucial for being able to interpret the results of this thesis. It suffices to say, that, for example in comparison to the simple addition of privacy using the basic composition theorem (see Definition 8), the accountant [47] can give a tighter bound for the privacy loss ϵ by using advanced composition and tracking statistical properties (higher moments) of the distribution of ϵ throughout sequential applications of the $\mathcal{M}$. The consequence of finding a tighter bound is, that smaller values for ϵ are achieved [47].

4.4 Model architecture and hyperparameters

Architecture changes relating to making the network suited for tabular data synthesis included, for example, swapping convolutional layers to fully-connected ones and choosing activation functions for the new layers that better suit the tabular case.

Architecture changes relating to making the network suited for tabular data synthesis included swapping convolutional layers to fully-connected ones and choosing activation functions for the new layers that better suit the tabular case. These choices for the new architecture were made based only on a limited number of tests. Less than five training runs with different seeds per architectural choice, such as the width or depth of the network, were tested. This was due to the focus on usability in this thesis, but also because training the whole model, especially in cases where large amounts of D networks were trained, was computationally expensive and time-consuming; the time it took to train a whole model with the hardware used (three NVIDIA RTX Titan GPU's) exceeded 12 hours in the worst case.

Because of the focus on usability and the considerable computational cost, no comprehensive hyperparameter optimization was done, and most hyperparameter settings were chosen based on the recommendations of Gulrajani et al. on training Wasserstein GANs [26], which is the source for the hyperparameter choices of Chen et al. [1] as well. In the following description of the network setup, if not stated

otherwise, the hyperparameter value choices come from this source.

The G network used is a fully connected network with two hidden layers, the largest being of size 256 with 16 outputs. The size of the noise vector z, fed in G to produce synthetic data samples, was set to 32 after testing on a few experiments at different sizes of powers of two. The activation function used was ReLU [35], except in the last layer of where a hyperbolic tangent (TanH) activation function, that outputs values ranging between $(-1, 1)$ was used. The choice of the TanH is due to the loss values of the WLoss possibly being negative or positive.

The D network is similar to a simple multi-layer perceptron classifier network with one hidden layer of size 128. Instead of a ReLU, as in the G network, a LeakyReLU [37] with α - negative slope value set to 0.2 is used in the hidden layers as in [26].

In both networks, batch normalization [52] was used based on a limited number of tests with- and without it. Batch normalization seemed to produce better results and make training converge faster. Similar tests, with less than five different models trained, were conducted for adding more hidden layers, but adding more did not seem to produce better results. Layer widths were picked based on testing with multiples of 2, up to a maximum of 1024. In addition, tests for using z-score standardization instead of min-max normalization (covered in the data section) were conducted, with the result of min-maxing seemingly generating better results in terms of this data and network architecture. As every iteration costs privacy, in a situation where no significant gains would have come from using a larger network, the decision was to use fewer trainable parameters to save iterations.

The gradient penalty hyperparameter was set to 10 according to the recommendation of Gulrajani et al [26]. The optimizer used with this implementation was the Adam [53] as in the GS-WGAN implementation [48]. The hyperparameter values used with Adam were those recommended in [26], that is $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 =$

0.9. The network weights for both G and D were initialized by values sampled from a normal distribution with mean 0 and standard deviation of 1.

4.5 Data

The Kaggle Cardiovascular Disease dataset [24] consists of 12 features and 70 000 observations. The dataset is used in this thesis to predict the presence of cardiovascular disease based on features such as age, weight, height, and systolic- and diastolic blood pressure. All features and the amount of observations affected by preprocessing are listed in 4.1. Table 4.2 contains statistical descriptives of continuous features after preprocessing and categorical and binary feature counts are depicted in Figure 4.3.

The main reasons for choosing this dataset were, that it is freely available making this work more reproducible and has features that make it a feasible proxy for common tabular EHR data. The feature to predict, presence of cardiovascular disease, is a common condition, and the prediction task could realistically be the goal of a medical data-analysis project. Also, this dataset has been used in other DP generation studies (see for ex. [54]). The features consist of mix of different types, as is often the case with tabular datasets: out of the 12 features, five are binary, two categorical, and five numeric.

Data preprocessing

As the features depict well-known and studied attributes (see Table 4.1), it was possible to use domain knowledge to detect outliers based on the plausibility of their values. For example, height was bounded to the largest reasonable value present in the data (210 cm), and blood pressure values were limited to a range of +-20 from values indicating a hypertensive crisis according to finnish national standards [55].

To avoid making the generation task easier, instead of median or mean-imputation,

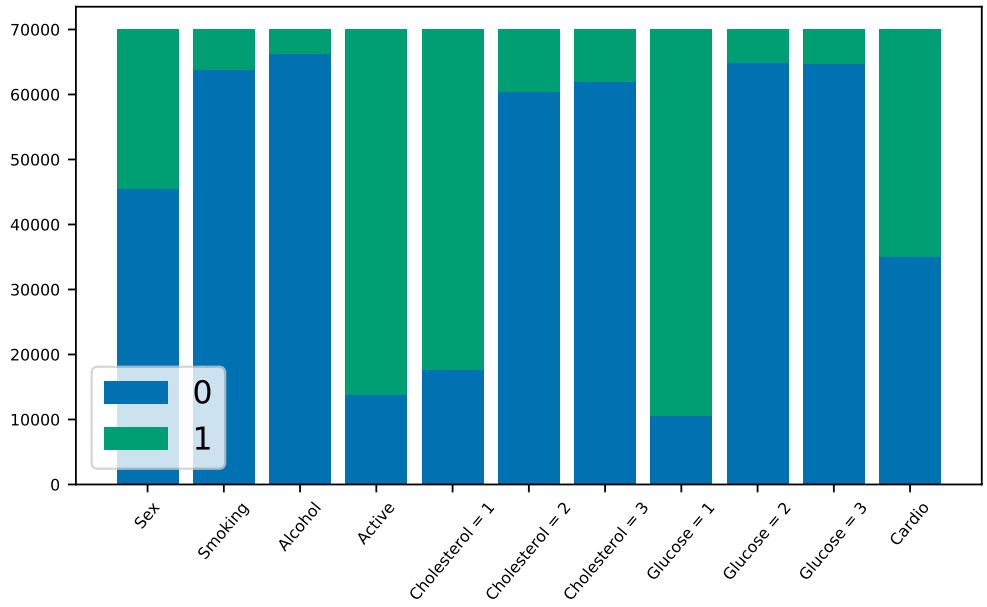

Figure 4.3: Frequency counts over binary and categorical variables. Categorical variables are split accross their levels.

KNN-imputation (see [56]) with $k = 3$ was carried out for outliers using the implementation from the scikit-learn package [57] (version 1.0.2). The two categorical variables 'cholesterol' and 'glucose' were one-hot encoded, after which all features were min-maxed, that is, scaled to $[-1, 1]$ to match the feature range of the G network output layer (TanH). This is an important step, as data given as input to D must be on the same range whether it is real data or sampled from G.

Feature	Type	Values affected by preprocessing
Age	numeric (days)	24
Height	numeric (cm)	24
Weight	numeric (kg)	77
Gender	binary	24
Systolic blood pressure (ap_lo)	numeric	312
Diastolic blood pressure (ap_hi)	numeric	1064
Cholesterol	1: normal 2: above normal 3: well above normal	24
Glucose	1: normal 2: above normal 3: well above normal	24
Smoking	binary	24
Alcohol intake	binary	24
Physical activity	binary	24
Cardiovascular disease (Cardio)	binary	24

Table 4.1: Features of the Cardio datasets and the amount of values affected by preprocessing. In case of binary or categorical variables, values that were affected by preprocessing were either outside the range of that feature or missing.

	Age (days)	Height	Weight	Ap_lo	Ap_hi
mean	19469	164.39	74.21	81.36	126.97
median	53.98	165.00	72.00	80.0	120.00
std	2466	7.98	14.37	9.63	17.07
min	14275	100	35.0	30.0	60.0
max	23713	207.0	200.0	190.0	240.0

Table 4.2: Descriptive statistics of continuous variables of the Cardio dataset after preprocessing.

5 Related works

The review of related works in this thesis is limited to synthetic data generation methods that use differential privacy, apart from a few mentions of closely related work that do not. There are works, that use GAN networks to create synthetic data, where the authors sometimes claim that their methods are privacy-preserving, even when DP guarantees are not given. However, even if the claims would be true, there is no way to verify or compare their results to those of, for example, this work, in a scientifically credible or meaningful manner. Federated learning approaches (see for example the federated learning version of GS-WGAN in [1]) are also left out of this thesis due to the focus on usability; including federated learning would make the scope too wide and discussing usability complicated. Federated learning approaches often require more complex setups and co-operation between different parties, whereas this thesis focuses on methods that could be used by a single institution or data holder.

The most important work related to this work is the 2020 article on GS-WGAN and its evaluation presented by Chen, Orekondy, and Fritz [1]. As explained in the methods section, this is the method modified in this thesis to suit DP tabular data generation. This review chapter focuses on comparing the GS-WGAN [1] to other similar work on synthetic data generation, motivating why modifying the approach to suit DP tabular data generation is worthwhile and what the theoretical interest behind this choice is.

5.1 Differentially private GANs

There are many Differentially Private GAN models trained with DPSGD [58], [59], [50], [54], [60], [61], [62], [63]. The GS-WGAN approach outperforms former methods that use the MNIST image [27] and fashion [28] datasets in terms of the tradeoff between the strength of privacy guarantees given and sample quality retained. As described in the methods section (see 4.3), the difference in performance to other methods can be attributed to the way only the information used by the private generator and not the D networks is sanitized, which makes it possible to exploit the synergy between the Wasserstein-loss function, gradient penalty and DPSGD [34] as well as the way the privacy amplification by subsampling technique [25] is applied to pre-training the D networks.

Starting our comparison to other works from a technical perspective and more specifically differences relating to how sanitation is performed, perhaps the closest works are those that use the Wasserstein loss function [26] and train the network using DPSGD, most notably the works of Xie [50], Zhang [59] and Frigerio [60]. The innovation of Chen et al. [1], that pretraining and training D can be done without additional privacy costs as D networks are not released [1] and the "privacy barrier" is placed between G and D sets it apart from these other works.

In the aforementioned works ([50], [60], [59]) perturbation is done during the training of the discriminator and they do not capitalize on the theoretical synergy between the WLoss, gradient penalty, and DPSGD (see Section 4.3). This results in the magnitudes of gradients being left unbounded with the consequence of estimates for the value of the clipping bound C in the DPSGD training being comparatively worse. With the GS-WGAN approach, the loss of information introduced by clipping is smaller, and an expensive hyperparameter search for C is avoided [1].

The other important distinction to other works is the way privacy amplification by subsampling (PABS) [25] is applied to DP GAN training. Perhaps the most

similar works in terms of the application PABS to that of the GS-WGAN [1] are PATE-GAN [61] and G-PATE [62], which are both a modification of the private aggregation of teacher ensembles (PATE) [64] method to suit GAN training. These networks also train multiple discriminators on mutually exclusive subsets of data, but applying DP in PATE is different from DPSGD and uses an ensemble-like voting mechanism to establish privacy guarantees. For example, in PATE-GAN [65] "teacher discriminators" are first trained on exclusive subsets and are then used to assign examples to discretized bins or categories via voting. An additional D network, a "student" network, is also trained and these are then used together to train the generator [65]. As an exception in comparison to the rest of the methods mentioned, the G-PATE [62], as Chen et al. [1] note "shares a close similarity" to GS-WGAN in that the D networks are also trained without privacy guarantees.

There are, however, disadvantages to PATE-based methods in that they show significant decreases in performance with high-dimensional datasets, as the authors of G-PATE [61] note. This may be related to the difficulty in discretizing the distributions, which is required for the ensemble voting to work well. Discretizing can be a challenging task when the number of dimensions is high and in terms of usability, it also brings the burden of choosing the number of bins algorithmically or otherwise. For this reason, to get strong ϵ values with high-dimensional data, it has been suggested, that dimensionality reduction methods may be needed with PATE-based GANs [62]. On the positive side, it has been suggested and observed by, for example, [1] that PATE-based methods might produce better results than GANs using DPSGD when the number of features is relatively low.

Usability

In addition to the aspects relating to privacy and sample quality, the method used in this thesis has usability benefits. As argued previously in this work, for a method

to be used in reality or on a large scale, it would have to be transferable between datasets, usable by a non-expert, and not require vast amounts of computational power. The GS-WGAN [1] provides solutions to these considerations to a large extent. First, no complicated pre-processing steps that require expertise are needed, such as was the case with possible dimensionality reduction and discretization required by the PATE-based methods. Second, due to how the gradient information is handled, privacy calculation is dataset independent (see section 4.3), and finally, no hyperparameter tuning in the case of the clipping bound C is needed.

Although the user is still left with some choices, such as the subsampling rate γ that determines the size of the mutually exclusive subsets during pre-training, there are far less optimizable parameters in comparison to many other methods. In fact, from a broader perspective, using a GAN, in general, can often also bring usability benefits due to GANs being able to avoid the need for the user to specify dataset related parameters, which other state-of-the-art DP synthetic data generation algorithms might require. For example the graph-based Private-PGM method [66] has achieved good results, but requires features to be made discrete, with discretization decisions preferably based on domain knowledge, literature, or some public dataset. Discretization is required in many other popular methods as well, such as Privbayes [67], based on Bayesian networks.

The use of public data or domain knowledge to, for example, pre-train or select parameters is something that has been seen with DP GAN methods also, but for other reasons than discretization. Examples of this can be found in less recent papers like the previously mentioned, 2018 published works of Yoon on PATE-GAN [64] or Zhang [59], where a public dataset is used to optimize the clipping bound C used in DPSGD. The requirement for public data is a problem as such data might not be available and is problematic in terms of usability as finding adequate data and making decisions on which possible data is suitable can be a labor-intensive task

that requires domain expertise. DP GAN methods, in general, do not involve such data-dependent decisions.

As with all methods, GANs do not come without disadvantages either. Some technical expertise and understanding in NNs is still needed in pre-processing (scaling and one-hot encoding) and in changing the input and output layer sizes. These steps, however, are arguably not very laborious nor do they require much domain expertise, or expertise relating to the method. Perhaps the most significant challenge relating to GANs is, that the amount of data needed compared to, for example, Private-PGM and Bayesian generation methods may be significantly larger. Also, other methods may give results that have less variance in quality between runs. For example, the number of iterations for the G to start learning effectively might vary significantly between training runs, affecting ϵ values. With this being said, from the viewpoint of usability, approaches using GANs still have a strong position among DP synthetization methods described in the literature.

Synthetic tabular data generation

To the writer's knowledge, neither the GS-WGAN [1] approach to gradient clipping nor the strategy in which PABS is applied to DP GAN training in it have been tested with tabular or tabular EHR data. This section covers in more detail (than as discussed briefly earlier) the questions that arise from this setup and links the questions to other works.

It is by no means self-evident that the state-of-the-art results of being able to retain high sample quality with $\epsilon < 10$ using MNIST [27] and fashion-MNIST [28] reported by Chen et al. [1] would transfer to tabular data. One reason for concern is that because image data is of the same type across its features and is often autocorrelated, it could be that using PABS with tabular data that is not, might lead to worse results. To be more specific, dividing data into subsets with tabular

data features of different types and possibly uncorrelated values could produce (on average) gradient vectors with larger magnitudes than images. If so, these gradients would then be more aggressively regularized by GP regularization, resulting in worse sample quality and perhaps also downstream classification ability of models in comparison to image data. If we suppose this line of thought would prove to be accurate, sampling rates equal to larger training data subsets for each D network could be needed because of the heterogeneity of the features, perhaps leading to a lesser benefit to be gained from this technique in the case of tabular data.

Closest to this work, especially in terms of how the tabular data synthetization results are evaluated are the work of Walia et al. [68] and the very recent work of Tao et al. [69]. To be specific, how assessment of sample quality is done in a qualitative manner by comparing real and generated feature distributions and correlation structure similarity between the synthetic and real datasets comes close to that of this work (see experiment evaluation details in Chapter 6.2). Unfortunately the work of Walia et al. [68] does not use DP but is still included as an exception because in addition to evaluation similarities the same dataset is used, and there are very few other articles similar to the work in this thesis. The inclusion of the work of Tao et al. [69] is also an exception in that it is still at preprint stage. The reasons for inclusion are similarity, broadness and the fact that the listed authors have done other highly regarded work on DP-synthetization (see, for example [66]). Tao et al. [69] compare many GAN, Bayesian and Marginal-based graphical methods such as the PATE-GAN [65] and DPGAN [50] discussed before as well as the Private PGM with MST [66] across several tabular datasets. They make an interesting interpretation of results, saying that GAN-based mechanisms fail to preserve basic data distribution statistics and that marginal methods fare better with most datasets and settings. There are, however, contradictory results to other papers suggesting that there might have been problems with training the GANs. For example, in the

work of Fang et al. [70] the DPCTGAN reached an AUC of 0.682 with the Adult dataset (both works use DPCTGAN and Adult), whereas Tao et al. [69] claim that "the synthetic data produced by GAN-based approaches yield classifiers that are generally no more accurate than a simple majority classifier," which is untrue based on other authors results.

Although none were found to use a similar method in terms of subsampling or the WLoss synergies with DPSGD, a few studies specifically target EHR or medical data generation. Perhaps the closest example is the recently published work by Torfi et al. [54] which focuses on tabular and time-series EHR-data. This work [54] uses the same Cardio dataset [24] and their approach, called RDP-CGAN [54] has an interesting architecture: a convolutional autoencoder that uses transposed convolutions with the aim of better capturing the dependencies between the features. While there are no visual comparisons, they present a similar comparison of downstream classifiers (See Chapter 7) with different ϵ values. While exact classifier accuracy values are not reported, a figure is presented with AUC values for the Cardio data with a classifier at $\epsilon = 10$, reaching approximately 0.73 with their method, an AUC ≈ 0.66 at $\epsilon = 1$ and AUC ≈ 0.61 at $\epsilon = 1$ (the real data AUC achieved in this work is ≈ 0.795). One of the other close examples in literature is the DP-CTGAN by Fang et al. [70] that also uses the Cardio dataset, achieving an AUC score of 0.6827 at $\epsilon = 2$. Neither of these works considers the GS-WGAN [1] in any way.

An another notable, but perhaps not of as high technical relevance to this work is the SPRINT-GAN [58], which focuses on patient-level clinical trial data sharing in a DP manner in the similar context of data related to blood pressure. This paper also presents an interesting and creative scheme for synthetic medical data evaluation where domain experts (physicians) score the realism of the samples generated.

6 Overview of models and experiments

This chapter presents an overview of the experiments and the differentially private synthetic data generation process. The pseudocode depictions in Algorithm 1 and Algorithm 2 give a detailed view of the pre-training and training processes of the main generator network. The downstream classification utility experiment is depicted in Algorithm 3. The code used will be made available as a github repository and can be gained access to by contacting the author.

6.1 Model training and synthetic data generation

The synthetic data generation process can be divided into three steps. First, the discriminator networks are pre-trained with mutually exclusive random subsets of the real data, according to the chosen amount of discriminators $ndis$ and subsampling rate, that is $\frac{ndis}{n}$. The subsampling rate is denoted with γ. This process is depicted as pseudocode in Algorithm 1. Second, the main G network, that is later published, is trained using the pretrained D networks. This process is described using pseudocode in Algorithm 2 2. Finally, synthetic data is sampled by feeding noise vectors $\boldsymbol{z}$ sampled from a standard normal distribution through G.

Algorithm 1: Pseudocode depiction of the discriminator pretraining process with privacy amplification by subsampling

Input: Dataset $\mathbb{X}$, subsampling rate γ, maximum number of pre-training iterations $iters_{pre}$, learning rates η_D, η_G, number of discriminator training iterations per generator iteration r, total number of discriminator-networks n_{dis}, batch size m

Output: $ndis$ of pre-trained D networks

Sample from $\mathbb{X}$, without replacement, subsets $\{\mathbb{X}_k\}_{k=1}^{K}$ where $K = ndis$, determined by the subsampling rate γ. Save list of indices relating to each subset to idx_k.

Initialize G_{temp} with parameters $\boldsymbol{\theta}_{G_{temp}}$

for k *in* $\{1, \ldots, K\}$ **do**

  Initialize discriminator D_k with parameters $\boldsymbol{\theta}_{D_k}$

  for $1, \ldots, iters_{pre}$ **do**

    for $1, \ldots, r$ **do**

      sample batch $\boldsymbol{b}$ of size m from data with indices in idx_k

      sample batch of noise $\boldsymbol{z}$ of size m where each $z \in \mathbf{z} \sim P_z = \mathcal{N} \sim (0, \sigma^2)$

      Update $\boldsymbol{\theta}_{D_k} \leftarrow \boldsymbol{\theta}_{D_k} - \eta_D \cdot \frac{1}{m} \sum_i \nabla_{\boldsymbol{\theta}_{D_k}} \mathcal{L}_{D_k} \left(\boldsymbol{b}, G_{temp}\left(\boldsymbol{z}\right)\right)$

    Update $\boldsymbol{\theta}_{G_{temp}} \leftarrow \boldsymbol{\theta}_{G_{temp}} - \eta_G \cdot \frac{1}{m} \sum \nabla_{\boldsymbol{\theta}_{G_{temp}}} \mathcal{L}_G \left(D(G_{temp}\left(\boldsymbol{z}\right))\right)$

Discard G_{temp}

The maximum number of pretraining iterations used in pretraining the models was 2000, the same as reported in [1]. In both the pretraining and the primary training process, the parameters of the D network participating in that particular step are updated multiple times per one update of G. The parameter that controls the amount of repetitions is $D_{repeats}$. In the experiments, the value $D_{repeats} = 5$ is used, based on a handful of experiments between values ranging from 1 to 15.

Primary training process

The primary training process of the differentially private generator is depicted in Algorithm 2. The most critical parameters affecting the total privacy cost ϵ are the noise scale σ, clipping bound C, the number of pre-trained discriminators $ndis$, training iterations T and the batch size m. The noise scale used was $\sigma = 1.07$ for all models as in [1]. All models were trained up to a maximum of 40 000 iterations. The batch size m was set to 32, based on experiments on a few different adjacent values of powers of two.

The data used to train models was the whole real dataset. This may seem unintuitive or wrong due to the risks of overfitting and falsely optimistic generalization results. However, due to randomly choosing between a large number of discriminators that see only their own mutually exclusive subset of the data, a decision was made not to leave data out of the training process.

The reasoning behind this decision is, that the subsampling strategy used causes some subsets of the data, especially with higher rates, to affect the training of G minimally or not at all. Additionally, the setup of GAN training, where G receives from D information only about the distance of the model distribution p_{model} and the current approximation of the empirical distribution $\hat{p}_{data}$ by D is not the same as, for example, training a classifier. This is due to the fact that the performance of G is not compared directly with ground truth, but only with the changing estimate by D. To even further back up this line of thought, it is worthwhile to mention that the perturbation to the gradients (GP, clipping and noise) regularizes the information flowing from D to G, which should also prevent overfitting. From a usability perspective and in the case of some real-world EHR data, which can be significantly smaller than the data used in this work, it can be argued that it would perhaps be overly cautious to leave part of the data out, considering these special characteristics. In order to avoid confusion, it is good to stress that the above only applies to

training the GAN models, not to the classification models used in the experiments where a train-validation-test split is used.

Algorithm 2: Train Model

Input: Dataset $\mathbb{X}$, subsampling rate γ, noise scale σ, clipping bound C, learning rates η_D, η_G, pre-trained discriminators D_k, K lists of indices corresponding to subsets of data used to pre-train the discriminator networks idx_k, the number of discriminator iterations per generator iteration r, number of discriminators n_{dis}, batch size m, maximum number of training iterations T

Output: Differentially Private generator G, total privacy cost ϵ

Initialize private generator G with parameters $\boldsymbol{\theta}_G$
for t *in* $\{1, \ldots, T\}$ **do**
  load a **randomly chosen** pre-trained discriminator D_k with parameters $\boldsymbol{\theta}_{D_k}$
  for i *in* r **do**
    sample batch $\boldsymbol{b}$ of size m from data with indices in idx_k
    sample batch of noise $\boldsymbol{z}$ of size m where each $z \in \mathbf{z} \sim P_z = \mathcal{N} \sim (0, \sigma^2)$
    update $\boldsymbol{\theta}_{D_k} \leftarrow \boldsymbol{\theta}_{D_k} - \eta_D \cdot \frac{1}{m} \sum \nabla_{\boldsymbol{\theta}_{D_k}} \mathcal{L}_{D_k} (\boldsymbol{b}, G(\boldsymbol{z}))$
  Update $\boldsymbol{\theta}_G \leftarrow \boldsymbol{\theta}_G - \eta_G \cdot \mathcal{M}_{\sigma,C} [\frac{1}{m} \sum \nabla_{\boldsymbol{\theta}_G} \mathcal{L}_G (D(G(\boldsymbol{z})))]$
  Accumulate privacy cost ϵ

Postprocessing generated samples

After training the model, the D networks are discarded, and the G network can be used to generate samples with DP guarantees. After the samples have been generated some postprocessing steps are needed: first the min-max scaling is reversed back to the original value ranges of the features. In this work, categorical variables were left one-hot encoded. This was a choice of only cosmetic importance and one-hot encoding can be reversed easily. As the outputs of G are not discrete in the case

of discrete or binary variables, the values produced are rounded to the closest full integer.

6.2 Experiments

In this thesis, the quality of the differentially private synthetic data generated is evaluated from two viewpoints: downstream modeling utility and similarity of the data in terms of sample quality (see research questions 1 and 2 in Section 1.3). These questions are investigated through comparisons between real data and different synthetic data generated by models with different levels of privacy guarantees achieved at different iterations of model training. Models are trained at the subsampling rates $\gamma = 1/250, 1/500, 1/750, 1/1000, 1/1500$. The term *statistical structure* is used to refer to the dependencies between data features as well as the characteristics of individual feature distributions.

This chapter is laid out so, that, first the details of the downstream classification experiment, a widely used evaluation method for synthetic data (see for ex. [1] [65]) are covered. The classification task is the binary classification of the presence of cardiac disease, represented by the target variable 'cardio'.

From examining the usefulness of the synthetic data for modeling, we turn to evaluations of the generated synthetic datasets statistical structure. Here the point of interest is to see whether the method can produce synthetic data with privacy guarantees and retain sample quality, that is, the statistical characteristics and structure of the data to an extent that it could be used for tasks where "realistic" samples are important, such as teaching or statistical aggregates.

6.2.1 Downstream classification experiment details

In the downstream classification experiment, models trained with synthetic data are tested on real data and compared to the results of a classifier trained and tested on real data. This experiment is, almost the same as that conducted by Chen et al. [1], with the only differences being that while Chen et al. use many different classification algorithms and, as a consequence, the hyperparameters optimized are different, while in this work only Logistic Regression [71] (LR) is used for evaluation. The only other classifier, with which no experiments were done, is a Random Forest (RF) ensemble [72] that was used for a "sanity check" of the suitability of using LR for the experiments as well as calculating feature importance scores, reported in Chapter 7, Table 7.1 for additional support.

Logistic regression (LR) is a supervised learning algorithm used to predict the probability of an event. As is the case with linear regression LR calculates a weighted sum of the input features X to determine the value of Y. The difference is, that it uses a logistic function to output a probability estimate of a variable taking on a discrete categorical value (in our case over the binary target variable 'cardio') instead of a continous real value. In this work, every LR-model is limited to using one of two regularization terms during training on different datasets: the $l1$ (lasso) or the $l2$ (ridge). Regularization aims to combat overfitting and affects how much influence individual features are allowed to exert on the model. The l1 term penalizes large absolute values of the regression coefficients, while the l2-term penalizes based on squared values of the model coefficients (for more information see for example [73]).

Classification accuracy is measured in this thesis with the rank-based Area Under Curve (AUC) metric [74]. The AUC value is an aggregate of a classifier's performance across multiple decision thresholds. A classifier with an AUC of 0.5 makes no better than majority class predictions, whereas a classifier with a value of 1 would be a perfect classifier. Unlike simple accuracy, the AUC is sensitive to class imbalance,

making it a more robust metric in cases where the size of the target variable groups may differ. Although the target variable 'cardio' of the dataset used is balanced as can be seen in Figure 4.3, this is a beneficial property in terms of this work as the target variable group sizes across synthetic datasets generated by different G may vary.

Hyperparameter optimization and experiment flow

Five private Generators were trained, up to a maximum of 40 000 iterations, using an amount of pre-trained D networks corresponding to the subsampling rates $\gamma = 1/250, 1/500, 1/750, 1/1000, 1/1500$, with the resulting mutually exclusive subset sizes seen by one D network being 180, 140, 93, 70 and 46, respectively. In addition, a non-private G network was trained to compare the effects of the generating process only.

Every model was saved once per 1000 iterations. In this work referred to as *checkpoints*, each of these saved states of the model are considered as individual models and were evaluated separately. Hyperparameter optimization for the models was also conducted separately for each checkpoint at different iterations. A stratified split was used for the synthetic datasets.

Hyperparameter optimization for the LR was conducted over the choice of the regularization term $l1$ (lasso) or the $l2$ (ridge), and the regularization strength. The regularization strength parameter c_{reg} is the inverse of the regularization penalty (where smaller values stand for stronger regularization). The parameter is inverted due to using the implementation of the scikit-learn package [57] (version 1.0.2). The 'liblinear' solver of the aforementioned package was chosen as it worked with both penalties and was seen to converge faster and to similar AUC values as other solvers available. The values of c_{reg} were randomly sampled from a logarithmic space between (0, 1) with 20 points tested with both l1 and l2 penalties.

The real and synthetic data used for this experiment were set up as follows. A split of training, validation, and test sets with size corresponding to fractions (0.8/0.1/0.1) were used for the real dataset. The resulting set sizes were 56000 for the training set and validation and 7000 for the test set. Synthetic data were sampled from each checkpoint model for a total of 63 000 synthetic data points, which were split with stratification over the target variable 'cardio' into two sets: a training and a validation set of the same size as in the real data case. The AUC values reported in the results for the synthetic-data trained models were obtained by testing with a real data test set sampled randomly for each different synthetic data evaluation. This is also why the synthetic data does not need to be split to three sets.

A total of 287 full model selection runs (40 checkpoint models, one per 1000 iterations for each of the 6 different γ settings and the real versus real baseline case) consisting of hyperparameter optimization and evaluation with the best hyperparameters settings were conducted. In this case, due to the significant amount of runs and considerable size of the dataset the possibility of conducting k-fold cross-validation was ruled out, but should be considered in the case of using this method with smaller datasets.

For clarity, the whole experiment procedure is depicted step-by-step with pseudocode in Algorithm 3. The pre-training of D networks and differentially private G training algorithms referred to in Algorithm 3 can be found in the Methods section (see Algorithm 1 and Algorithm 2).

In the results and the pseudocode depiction of the process, models are denoted so that, for example, M_{250}^{10k} refers to a 'checkpoint' model at 10 000 iterations and a subsampling rate of 250. The privacy parameters were held constant across all these models at $C = 1$ and noise scale $\sigma = 1.07$ as in the work of [1]. The model denoted $M_{baseline}$ is a G model trained with no privacy techniques applied and with one D

only (no noise or gradient clipping and $\gamma = 1$). This model is used as a baseline for comparison. In all experiments, RDP parameters that equate to a value of 1e-5 of the delta privacy parameter were used (see Definition 10).

The logistic regression coefficients and the results of a random forest ensemble classifier [72] feature importance measures in the case of a classifier trained and tested on real data are reported in table 7.1. This supplementary knowledge was added to aid with interpreting the results; the knowledge of which features a classifier uses on the real data to separate the classes from each other helps evaluate how the structure of the synthetic data relates to the AUC scores of the classifiers with different privacy settings.

Algorithm 3: Downstream Classification Quality Experiment procedure

Create sets $h \in \mathbf{H}$, where $\mathbf{H}$ denotes combinations between 20 values of c_{reg} randomly sampled from a logarithmic space between (0, 1) and regularization term choice of either $l1$ or $l2$

for γ *in {1, 1/250, 1/500, 1/750, 1/1000, 1/1500}* **do**

  Pretrain all K (denominator of γ), networks D_k^γ for model M_γ as in Algorithm 1.

  train (M_γ using D^γ) as in Algorithm 2 and save "checkpoint models" $M_\gamma^{1k}, M_\gamma^{2k} \dots M_\gamma^{40k}$ every 1000 iterations

for γ *in {1, 1/250, 1/500, 1/750, 1/1000, 1/1500}* **do**

  for i *in {1k, 2k, . . . , 40k}* **do**

    split dataset $\mathbb{X}$ of size n with stratification by the Y variable 'cardio' to $\mathbb{X}_{train}$, $\mathbb{X}_{val}$, and $\mathbb{X}_{test}$ with proportions 0.8/0.1/0.1 * n respectively.

    sample s $= 0.9 * n$ synthetic data points from M_γ^i and split with stratification to $synth_{train}$ and $synth_{val}$, where size of $synth_{train} = real_{train}$ and size $synth_{val}$ = size of $real_{val}$

    for $h \in \mathbf{H}$ **do**

      train classifier $LR_{validation}$ with hyperparameters h and data $synth_{train}$

      evaluate $LR_{validation}$ against $synth_{val}$.

      save best h in h_{best}

    train classifier LR_{test} with h_{best} and data $synth_{train}$ combined with $synth_{val}$

    evaluate LR_{test} using $\mathbb{X}_{test}$

    save the result of the downstream classification utility test for M_γ^i

    empty the set h_{best}

6.2.2 Evaluating sample quality

A comparative assessment of the effects of different privacy levels on the method's ability to retain sample quality (see research question 2 in Section 1.3) in this thesis consists of three comparisons between models trained with the different privacy settings and the real data. These are comparisons of correlational structure using Spearman's rank correlation coefficient (see for example [75]), a visual examination of the change of the continuous feature distributions, and a visual examination of the change in binary and categorical variable distributions.

The sample quality results are presented in the figures 7.2, 7.3, 7.4 and for three differentially private Generators at subsampling rates $1/250, 1/500, 1/1500$, and the 'baseline', non private G $M_{baseline}$ and real data for comparison. The models for this evaluation were chosen based on their performance in the downstream classification task; the models presented are the ones that achieved the best AUC to ϵ tradeoff from the models trained with the same subsampling rate γ.

7 Results

This chapter presents the results, laid out as follows: first, the downstream classification utility experiment results are presented in Section 7.1. The details of this experiment can be found in the Subsection 6.2.1 of the previous chapter. Algorithm 3 contains a pseudocode depiction of the experiment procedure in full and hyperparameter optimization details for the Logistic Regression classifier are covered in the Subsection 6.2.1. The sample quality evaluation results are presented starting from Section 7.2. The Subsection 6.2.2 in the Experiments chapter describes the evaluation in more detail. Discussion is separated to a chapter of its own (see Chapter 8).

7.1 Downstream classification utility

Figure 7.1 compares results of the downstream classification utility experiment with multiple synthetic datasets sampled at different checkpoints (model at different number of iterations) from five differentially private models with different subsampling rates. The models are denoted M_{250} for $\gamma = 1/250$, M_{500} for $\gamma = 1/500$, M_{750} for $\gamma = 1/750$, M_{1000} for $\gamma = 1/1000$ and M_{1500} for $\gamma = 1/1500$ and are compared to a non-private Generator denoted $M_{baseline}$. The resulting sizes of the mutually exclusive subsets of data for each D were 180 for M_{250}, 140 for M_{500}, 93 for M_{750}, 70 for M_{1000} and 46 for M_{1500}. In all cases, the discriminators D were pretrained for 2000 iterations prior to training the main G from which synthetic data is sampled.

	Ap_hi	Ap_lo	Cholesterol = 1, 2, 3	Age	Weight
LR Coefficient	7.57	1.32	-1.42, -1.02, -0.34	1.30	1.81
RF Importance	0.47	0.19	0.07, 0.02, 0.10	0.09	0.04

Table 7.1: Logistic regression (LR) coefficient values and feature importance scores of a random forest classifier, both trained on real data. Coefficient values are presented for the five most important features sorted according to the LR model coefficient sizes.

To aid interpretation, logistic regression (LR) coefficients for five features with the highest effect on a model trained on real data are presented in the table 7.1. For additional support and to account for possible method-dependencies of choosing LR, feature importance scores obtained from a random forest classifier with 50 estimators and a maximum depth of 5 have also been added to the table.

The best overall results in terms of the ϵ, AUC tradeoff were achieved with synthetic data sampled from the model M_{1500} with $\epsilon = 6.45$ and AUC = 0.717. Compared to the real data LR model AUC of 0.795, the difference was 0.078. Accounting for the loss of information caused by generating data with any model, that is, when compared to the results obtained with data sampled from the non-private $M_{baseline}$ (AUC = 0.788) this difference drops to 0.071.

Other private models required more than double the privacy budget to reach classification accuracy similar to M_{1500} with the closest model, M_{750} achieving AUC = 0.715 at $\epsilon = 13.9$. The next best tradeoff was obtained with data sampled from M_{1000}, which ultimately failed to reach the same AUC, with its highest score being 0.687 at $\epsilon = 14.7$.

In general, models with weaker privacy guarantees and a smaller subsampling rate were able to reach higher values of AUC eventually, but at high privacy costs. In comparison to M_{1500} that reached the best tradeoff, for example, M_{500} reached

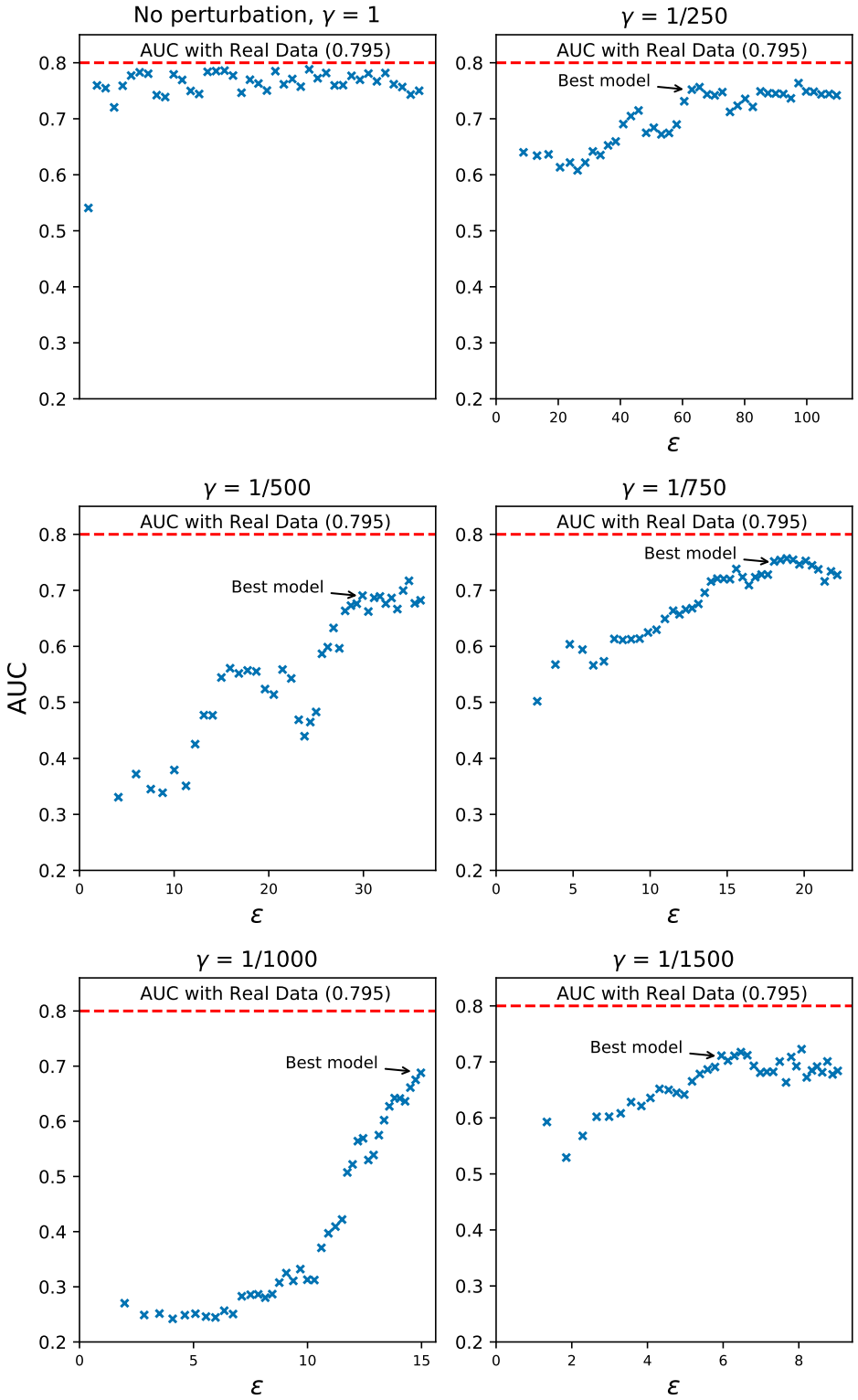

Figure 7.1: Downstream classification task results comparing AUC and ϵ with a non-private model $M_{baseline}$ (no gradient clipping or noise) and models with differential privacy using a noise scale of $\sigma = 1.07$ and a clipping bound C of 1. All models, regardless of subsampling rate γ are trained to a maximum of 40 000 iterations. Each marker represents a 'checkpoint' at every 1000 iterations, where a synthetic dataset was sampled and experimented with. The model at a checkpoint with the lowest ϵ to AUC ratio close to the highest AUC of the checkpoint evaluations for a specific model is chosen as the "best model". The visualizations of distributions in Figure 7.2 are from data generated by these "best" models.

		Age (days)	Height	Weight	Ap_hi	Ap_lo
Real Data	mean	19469	164	74	126	81
	std	2466	7.98	14.37	17.07	9.63
$M_{baseline}$	mean	19259	161.0	90.0	128.0	92.0
	std	1485	8.0	14.0	17.0	10.0
M_{250}	mean	18291	163.0	108.0	117.0	89.0
	std	1063	7.0	16.0	14.0	10.0
M_{750}	mean	18734	159.0	106.0	117.0	97.0
	std	1056	10.0	12.0	15.0	11.0
M_{1500}	mean	19169	151.0	113.0	126.0	88.0
	std	1414	10.0	19.0	14.0	13.0

Table 7.2: Means and standard deviations of continuous features calculated from synthetic datasets sampled from models performing best in the downstream classification utility task at different subsampling settings. The data are the same as used in Figure 7.2

the value 0.717, close to that of M_{1500} at $\epsilon = 34.8$, nearly six times more. The best AUC value obtained with privacy-preserving models was reached by M_{250} at an AUC of 0.752 with $\epsilon = 63.0$.

7.2 Sample quality

Continuous features

Figure 7.2 shows distributions of continuous features generated by models performing best in the downstream classification task at settings M_{250}, M_{750} and M_{1500}, together with samples from the non-private $M_{baseline}$, and the real data distributions. Note that the y-axis density value range varies to provide better resolution for each variable. Table 7.2 shows means and standard deviations for the distributions in Figure 7.2.

The results show that in comparison to real data continuous feature distributions, the method smooths the distributions even when trained without privacy guarantees. Looking at Table 7.2 we can see that for features with a high standard deviation (SD) in the real data, SD values tend to shrink in the synthetic datasets. Looking the distributions shown in Figure 7.2, these (high SD) features, such as age, seem to have their mass "compressed" towards the mean. On the other hand, distributions closer to a normal curve and lower SD, such as 'weight' or 'height', are affected less, even without perturbation. Considerable x-axis shift in comparison to the real data distributions is also present. An example of this can be seen in the case of the feature 'weight'.

Correlational structure

Figure 7.3 shows a comparison between Spearman Rank correlation [75] coefficient values calculated between the continuous features across the same best performing models shown in 7.2. Significant correlations are marked with (*) for a significance level of $p < 0.05$ and (**) for $p < 0.01$.

Even in the case of the non-private synthetic data sampled from $M_{baseline}$, many of the dependencies in the real data between features are lost, as is the case with, for example, the correlation between 'weight' and 'ap_hi'. In addition, the synthetic datasets, especially those sampled from private models show new correlations that are not present in the real data.

Binary and categorical features

Figure 7.4 compares samples from the same sources as the continuous feature comparison (see Figure 7.2) before, but in terms of the binary and categorical feature distributions. The non-private synthetic data generated by $M_{baseline}$ appears to capture the distributions of the real data well, but when DP is applied, there are

considerable deviations from the real data case. This is especially true for data sampled from M_{1500} ($\epsilon = 6$). Data generated by the other private models with less strict privacy guarantees and the non-DP $M_{baseline}$ appear more similar to each other and the real data. In the case of features where the number of positive cases is low to begin with, such as 'alcohol', adding DP seems to often further decrease the amount of positives. For the categorical features 'cholesterol' and 'glucose', stronger privacy guarantees such as in the case of M_{1500} seem to also balance the size differences between the counts.

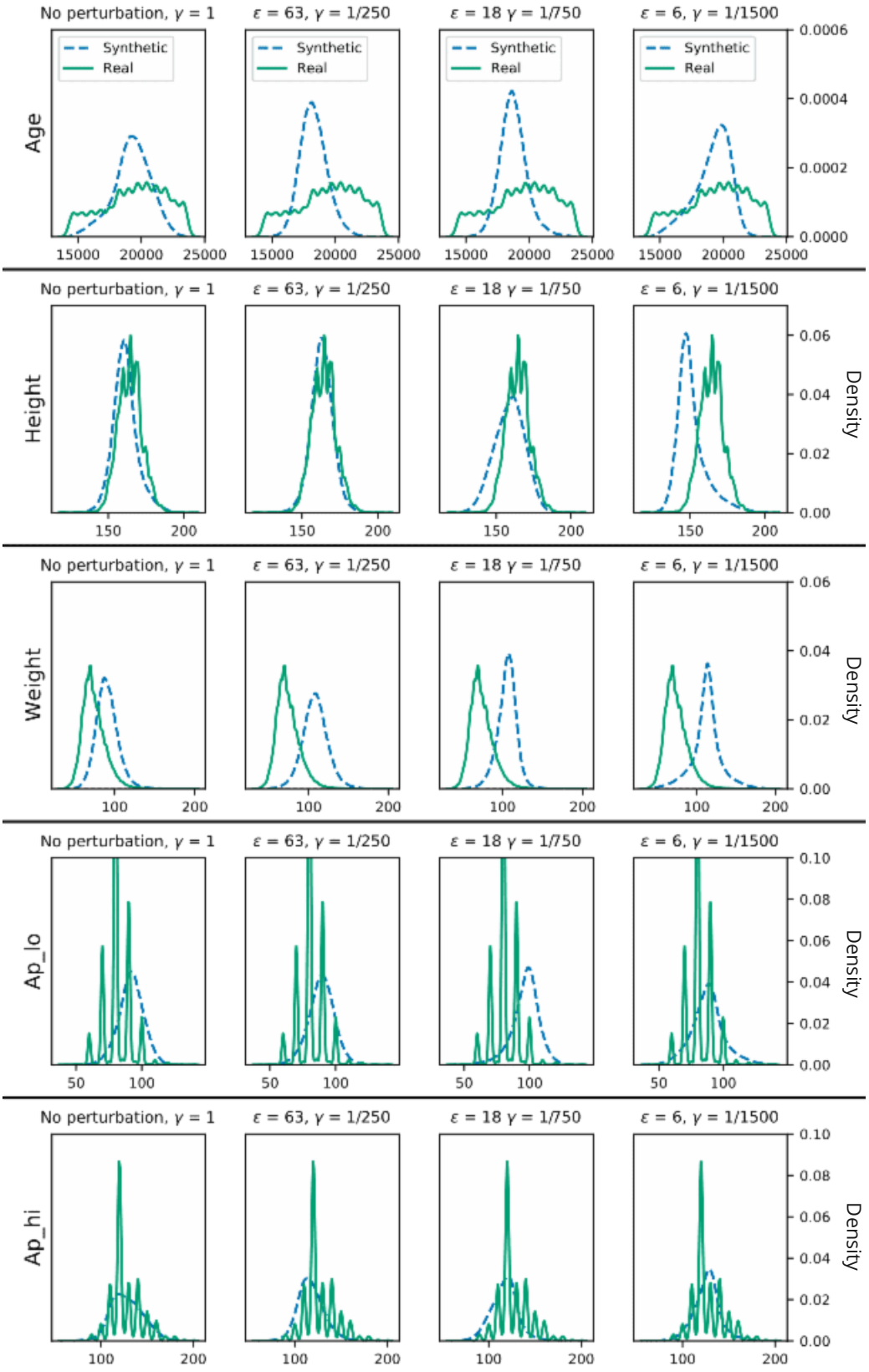

Figure 7.2: Comparison of distributions of continuous features of synthetic data produced by the best performing models of the downstream classification task based on AUC to ϵ tradeoff. Note that the y-axis varies in range. Real data distributions are included for comparison. Columns are different models at different sampling rates γ and ϵ values whereas rows are the different continuous features.

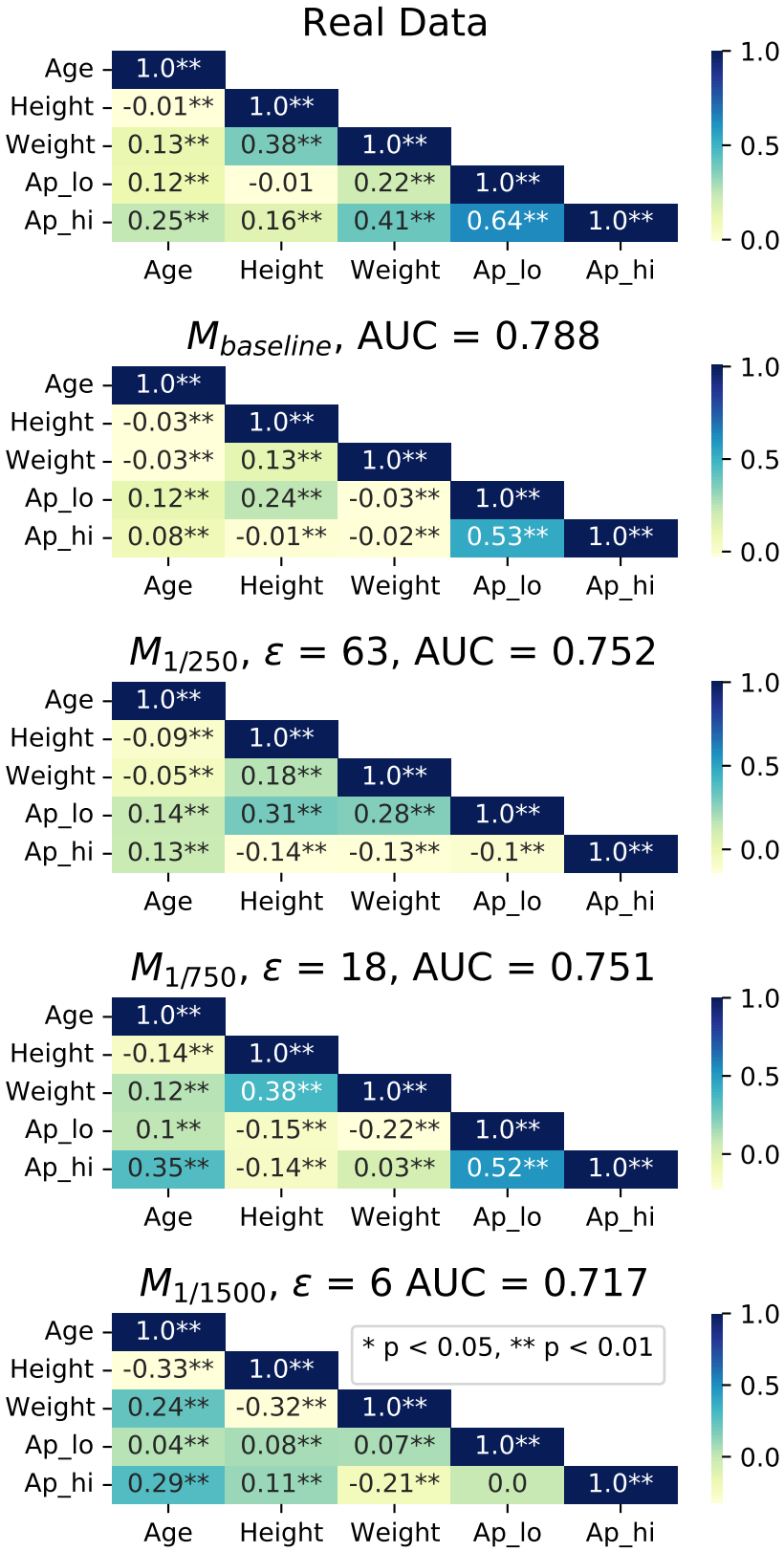

Figure 7.3: Spearman rank correlation coefficients of continuous variables compared between synthetic and real dataset across three of the best performing private models in the downstream classification utility experiment. The synthetic data generated by the non-private $M_{baseline}$ as well as the real data case are included for comparison. One asterisk (*) denotes a significant at a p-value < 0.05 and ** marks significance at level < 0.01. 7.3

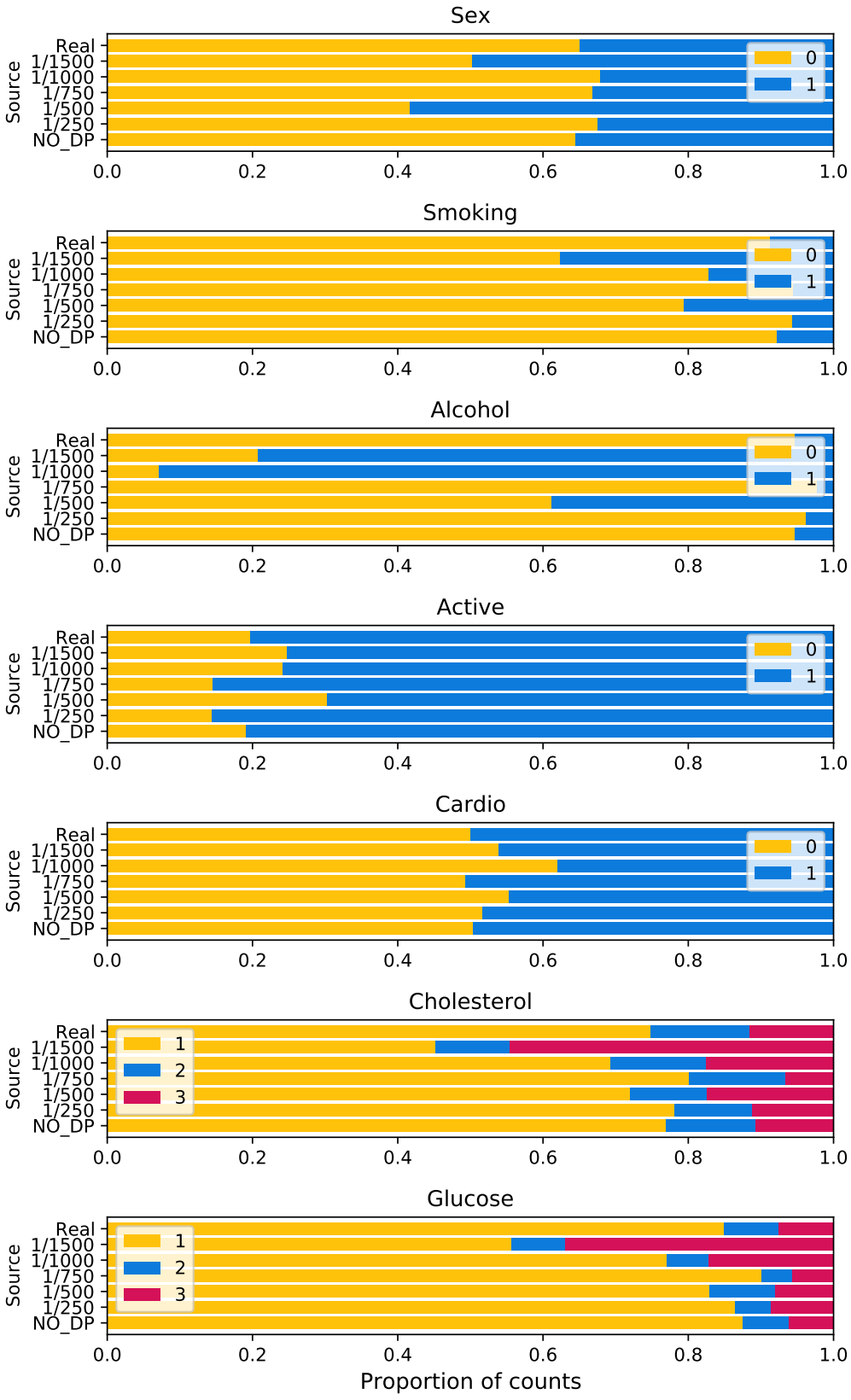

Figure 7.4: Comparison of frequencies of binary and categorical features between data sampled from the best performing models in the downstream utility experiment. Real data distributions are included for comparison.

8 Discussion

Before discussing the results, a few words and reminders of what should be considered when interpreting GAN models' products are in order. For a refresher on specifics of GANs see Section 2.2.

The results in this thesis are based only on a limited number of possible samples from a limited number of models and should be viewed as such. This is to say, that the findings can not be extrapolated over all possible samples obtainable with the method, and some other settings might have produced better results. GAN models are known to have large variance in what parameters they converge to between training runs (see for example [49]). Consequently, the synthetic data sampled from different generators can be significantly different with regards to some features, and it cannot be known, in light of this work, which of the minima some models have converged to would be "best" in general. Especially the sample quality evaluation results may be highly dependent on model-specific convergence as no comprehensive tuning for GAN model parameters was done. Because of this, the results of the sample quality evaluation should be seen only as illustrations of possible effects of applying the method to synthetic data generation.

With this being said the absence of comprehensive tuning can also be seen as a strength regarding the focus on usability in this work; the qualitative evaluations are perhaps more in line with what would be feasible if this method was used in a real-life setting. A comprehensive and computationally expensive optimization to

attain the near 'best possible' synthetic data could have shown results unattainable for potential non-expert users. With these considerations in mind, let us now turn to discussing the results.

8.1 Downstream classification utility

In the downstream classification utility task (see Figure 7.1) the private models, especially those with higher subsampling rates, required more training iterations to obtain reasonable AUC values. The benefits to privacy attained by applying privacy amplification by subsampling (PABS), however, outweighed the costs caused by more training iterations. This suggests that the benefits seen with the use of PABS with image data by Chen et al. [1] can also be reaped when tabular data with mixed feature types is used.

The size of this benefit can be seen to have been significant. For example, M_{1500} needed close to 20 000 iterations to show saturation, meaning more iterations did not cause AUC values to rise significantly, but values of ϵ were still lower than eight at this point. Let us illustrate the difference between settings with a simple ratio of "efficiency", that is $\frac{\text{AUC}}{\epsilon}$, taking into account checkpoints where the model AUC was > 0.5 and had not yet clearly saturated. Looking at this ratio for the different subsampling settings we can see that there are large differences between how much AUC improvement was gained per one unit of ϵ: the ratio was 0.011 for M_{250}, 0.020 for M_{500}, 0.051 for M_{750}, 0.047 for M_{1000}, and 0.11 for M_{1500}. The increase from M_{250} to M_{1000} is almost tenfold, which, even though this ratio is not perhaps a very valid metric, significant.

Even higher subsampling rates could perhaps have been experimented with, although, if we cross-check the results of the same models in the downstream utility experiment against the sample quality results, there are some possible signs that the subsampling rate of M_{1500} where the subset size seen by each D was 46, could be

close to the smallest one feasible with this data. This line of thought is supported by the observation, that especially in terms of the categorical- and binary features (see Figure 7.4) the samples produced by M_{1500} start to show much larger deviations compared to the models with lower subsampling rates, even though the AUC value stays similar.

One probable explanation for why the AUC does not differ more although the samples are significantly different accross many features is that the models were selected based on downstream utility task performance and the features, such as 'ap_hi', that were important for classification (see Table 7.1) were preserved better by the models than many of the other features. If this assumption was to be true, it would have negative implications for the continuation of using downstream classification utility as a metric (see for ex. [1], [54]).

To be more specific, it is possible, that the use of classifier performance as a quality metric for synthetic data leads to a biased and limited view of synthetic data quality, as it favors choosing models that have converged to parameters that retain the distributions of features important in classification, while other features can be very dissimilar in comparison to real data. In other words, you get what you measure, and synthetic data chosen this way is perhaps usable only for the classification task and the feature it targeted.

In terms of interpreting these results it is also good to observe, that in the classification task of this thesis, the features' importance for classification was divided very unevenly between the variables (see Table 7.1), which could make the classification task easier in this particular case. With some other data, where the most important features in terms of classification would, have high standard deviation and/or a lot of outliers, the effect of subsampling and application of DP could be much more detrimental to downstream classification utility. This is in reference to, cases like was seen to happen, for instance, to the distribution of 'age' (see Figure

7.2) becoming more concentrated at higher subsampling rates.

In comparison to other works using the same data, the 2022 results of Fang et al. [70], with the DP-CTGAN are similar, or perhaps better than those of this work regarding downstream classification utility, although direct comparison is difficult as extensive hyperparameter search was not performed for the models used in this thesis, and a different classifier model was used. Fang et al. report an AUC value of 0.6827 at $\epsilon = 2$ whereas the models of this work reach AUC values of ≈ 0.61 with a value of ϵ a little under 2 in the best case. The DP-CTGAN uses a conditional GAN setup, but does not include PABS in training.

The classification accuracy obtained by the convolutional RDP-CGAN of Torfi et al. [54], reported only at $\epsilon = 10$ and in a figure, were, based on a visual approximation, between AUC ≈ 0.72 and ≈ 0.73. Their results come close to those of this work, but fall perhaps a bit short as the model M_{1500} reached similar values of AUC already at a smaller value of $\epsilon = 6.45$, AUC ≈ 0.72. The RDP-CGAN [54] did not use PABS either.

8.2 Sample quality

The phenomenon of values concentrating towards the mean when stronger privacy guarantees are in effect is interesting not only in terms of the downstream classification utility but also in terms of the sample quality results as well. Specifically, looking at Figure 7.2, it would seem that distributions are affected significantly by the generation process itself, even when no perturbation is applied. This is probably a result of the gradient penalty (GP) regularization pushing the gradient vector magnitude closer to one during training. In terms of the synergy between the Wasserstein-1 loss, DPSGD, and GP (see Section 4.3), one could interpret this "compression" of feature values as a way to transfer some of the information loss due to clipping in DPSGD to the model. In other words, if distributions with a higher

standard deviation are more affected by clipping and vice versa, one could see GP as "preparing" the values of the features for DPSGD. This presents an interesting viewpoint into applying DP to synthetic data generation via GANs: it would seem that there are, and could be more beneficial ways to getting better tradeoffs between privacy and synthetic data quality by transferring some of the perturbation to the model or preparing features for DPSGD. Despite the benefit, sample quality is still lost before clipping or after it, and there probably is a limit to how much can be achieved with technical solutions such as this, as in the end, it is part of the idea of differential privacy to change the distributions.

The sample quality results are consistent with previous reports of training machine learning models with DPSGD having a similar smoothing and compressing effect where observations come closer to the mean, and it is also known that this compression is done at the expense of observations that have feature values far away from the mean or in the "tails of the distribution" as put by Suriyakumar et al. [76]. To be more specific, in their work [76], it was observed that when DPSGD is used to train classifier models to predict group membership, stricter privacy guarantees led to making more majority group predictions at the expense of minority groups that had more feature values in the tails. This effect of DPSGD training could also be behind the more drastic effect the method used in this thesis has on continuous distributions, where values are more spread out, as was seen with, for instance, the feature 'age' (see Figure 7.2).

For the continuous variable distributions also the x-axis shift visible in Figure 7.2 grows perhaps slightly larger when stricter privacy settings are set, which would imply that this does not happen because of architectural choices. This interpretation is, however, based only on a visual examination and is not evidence to make definitive conclusions about the source of the shift. Nevertheless, if we assume this effect exists, it would perhaps be even more detrimental to sample quality, than the compression

towards the mean, especially when viewed from the point of view of this methods ability to generate samples for uses where retaining realism is important.

Speculating further, it would be plausible that this shift is due more to the amplification by subsampling than DPSGD. This is because values closer to the mean should have a relatively stronger effect on training in DPSGD, especially when training progresses. The observations that deviate from other observations should eventually have larger and larger gradients, which of a more significant proportion then gets clipped. In other words, values close to the mean should be emphasized by DPSGD, and as such, the x-axis shift being due to DPSGD seems improbable. On the other hand, too small subsets due to a high subsampling rate could well shift the distributions. This implication is something to keep in mind when creating synthetic data for applications where realistic samples are vital using models that take advantage of PABS.

Turning to the binary and categorical features (see Figure 7.4), between the DP models M_{250}, M_{500}, M_{750}, and the non-private $M_{baseline}$, most distributions seem to be quite similar across the synthetic datasets and retain the distributions seen in the real data. With that being said, perhaps the most interesting results are the size of changes in data generated by the models at the two most strict subsampling settings M_{1500}, ($\epsilon = 6.45$, AUC $= 0.717$) and M_{1000} ($\epsilon = 14.7$, AUC $= 0.687$). These models produced very different feature distributions in comparison to the real data and the other models. For instance, in the case of the feature 'alcohol', which did not have a large impact on classification (see 7.1), the distribution in data sampled from M_{1500} and M_{1000} was nearly the opposite of that produced by the model with the next strictest subsampling rate M_{750} $\epsilon = 18$ AUC $= 0.715$. Distributions that were important in terms of classification were, however, in some cases, very different too. This can be seen, for instance, in the distribution of 'cholesterol' of the synthetic data sampled from M_{1500}.

Although the cause of these changes can not be inferred from these experiments, by examining the Figure 7.4 it would seem plausible that this phenomenon has to do with the relationship between the size of the subsets of data the D networks see, and the frequency of values in the real data for those features. With both the 'smoking' and 'cholesterol' features, the number of positive cases were low in the real data; for 'smoking,' there were only approximately 1/10 positive cases, and for 'cholesterol', both the categories 2 and 3 were much smaller than the category 1. One explanation could be that at higher subsampling rates, individual D networks see very few and sometimes none of these less represented examples, leading to difficulties in learning the distribution. Furthermore, as GAN models produce only continuous values, that are then rounded to the closest value for that feature, it could be that what is seen with these features is the estimate often being close to the class-boundary and more cases falling in some category by chance. These results further show that two synthetic datasets that obtain similar AUC values can have drastically different dependence structures among variables and support the idea discussed, that using classification utility as a measure for DP synthetic data quality may lead to synthetic data biased towards the target of that task.

The dependency structure comparisons, measured with the Spearman rank correlation coefficient shown in Figure 7.3 paint a similar picture as the previous sample quality results: a synthetic dataset that may work for some classification task and model development may have little to do with the internal structure of the real data. The AUC value can be similar between data sampled from models that are trained with privacy constraints, while the correlations between even the most influential features like 'ap_hi' and other features change and relationships that should hold for synthetic data in order for it to be realistic are broken.

In some cases, these relationships can even change to be opposite. For example, data sampled from the model $M_{1/1500}$ has a correlation of (-0.32) between 'weight'

and 'height' whereas it is (0.38) in the data sampled from $M_{1/750}$ and 0.13 in the real data. This might be related to the choice of models based on the downstream classification utility experiment and the possible bias that may be induced by this manner of model choice as discussed before. All correlations were seen to hold or strengthen their statistical significance in the synthetic data even as they change in direction and magnitude. This is probably due to the size of the data and the "compression" of data towards the mean as discussed before. Based on these results, retaining a realistic correlational structure with this method can prove to be very challenging, especially with lower values of ϵ.

9 Conclusion

The aim of the research presented in this thesis was to create differentially private synthetic tabular data by modifying the GS-WGAN method presented by Chen, Orekondy, and Fritz [1] to suit this purpose. The research questions (see section 1.3) were, whether synthetic data with strong privacy guarantees that would retain both 1) sample quality and 2) downstream modeling utility could be created, and 3) whether the privacy by subsampling (PABS) technique, as applied by Chen et al. [1] could be used to gain similar privacy benefits with tabular data as have been observed [1] with image data. To the authors knowledge, this is the first work in which this manner of PABS application to DP-GANs has been done with tabular data. The technical choices in this work focused on usability in a real-life scenario.

The results show that the method could be modified to suit tabular data containing mixed feature types while preserving downstream utility and strong privacy guarantees. Privacy loss in the best case was as low as $\epsilon = 6.45$ at AUC $= 0.717$, in the downstream classification task, while relatively little accuracy was lost, with the AUC value decreasing by 0.078 in comparison to the real data AUC value of 0.795. However, while downstream utility was preserved, retaining sample quality and producing realistic samples proved to be a more difficult. Data sampled from models that obtained good AUC values with strong privacy guarantees showed significant deviations in correlational structure, binary and categorical value distributions, and there was considerable x-axis shift coupled with a compression of density towards

the mean in the case of continuous feature distributions.

From a broader perspective, two findings in this study are especially interesting for in terms of the field of privacy-preserving synthetic data generation and DP-GAN modeling. First, the application of PABS to discriminator training enabled significantly stricter privacy guarantees with tabular data. This suggests that future work should consider this technique also with other than image data. Not taking advantage of this technique may mean losing out on easily attainable privacy benefits, as it is plausible that with even small datasets, splitting data into at least a small number of subsets would not cause harm that the benefits would not outweigh. Regarding the subsampling rate, the results of this thesis suggest that for applications where downstream utility is important, data can be split to perhaps a larger amount of subsets. However, if retaining realistic samples is a priority, one should be more careful when applying PABS.

Second, the results raise important questions about the widely used downstream classification utility as a way of evaluating synthetic data quality. The AUC staying similar between datasets with very different and non-realistic samples could indicate, that training models focusing on classification utility can result in synthetic data samples biased towards that task. Future work should investigate whether separate synthetic datasets should be created for specific uses. It is probable that creating "general" DP synthetic data that has utility for many tasks in the way real data does can be extraordinarily difficult and perhaps not necessary. It is also worthwhile to question whether the goal of creating such general datasets can be wasteful with regard to differential privacy; why split a already limited budget?

In a broad sense, it can be said, that the advancement of synthetic data generation would benefit from new metrics that would be more unified and would take into account dataset specificity, for example, through evaluating the effects of feature importance in downstream utility tasks and tracking loss of sample quality.

In addition, a metric that would allow to compare different algorithms for creating DP synthetic data, for example, by describing the amount of information that is extracted from the data per privacy loss unit spent would be beneficial. These are, of course easier, to imagine and talk about, than to devise.

From a more technical point of view, there are solutions that could have been better, and that could be useful to investigate in future works. One essential point is, that as pretraining is free of privacy cost with this method, investigations into finding the optimal number of pretraining iterations for different cases could lead to fewer iterations needed. A limitation related to this with regards to the results is that some models might have had problems beginning to learn efficiently, as was perhaps the case with the model M_{1000}, which could have reached better AUC values on a different training run.

In addition, regarding GAN modeling, better pre-processing and architectural choices in this thesis might have led to better results. For example, using a convolutional autoencoder inside the GAN architecture such as in the work of Torfi et al. [54] could be a solution that might benefit capturing the internal structure of many different types of data and enable training with fewer iterations. This idea is backed up by the fact that Torfi et al. [54] reported similar AUC values with the same dataset, but at slightly higher ϵ than in this work, but did not use PABS, which might enable their model to reach better tradeoffs. Alternatively, combining PABS and pre-training the D models with the DP-CTGAN presented by Fang et al. [70], which seemed to outperform the models in this work, could lead to even better results.

Related to these considerations of how to represent information, whether through architectural choices or other means, it has been discussed in the literature (see for ex. [77]), that a part of achieving better privacy guarantees with DP methods might largely be a problem of feature engineering and not so much a question of what kind

of method is used. It is, however, the belief of the author that if a breakthrough or "AlexNet moment" is to come for differentially private data sharing, it will be a mosaic put together from small improvements and realizations made as a part of investigations into many different techniques. As of now, there are still, without doubt, many important findings to make across the different lines of research in the field, regardless of differences in approaches.

References

[1] D. Chen, T. Orekondy, and M. Fritz, "GS-WGAN: A gradient-sanitized approach for learning differentially private generators", *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 673–12 684, 2020.

[2] Finnish Ministry of Social Affairs and Health, *Act 552/2019 on the secondary use of health and social data*, 2019. [Online]. Available: `https://stm.fi/en/secondary-use-of-health-and-social-data`.

[3] European Commission, *Act 2008/594/EC, Commission recommendation on cross-border interoperability of electronic health record systems*, 2008. [Online]. Available: `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32008H0594&from=EN`.

[4] D. S. Dhami, A. Soni, D. Page, and S. Natarajan, "Identifying Parkinson's patients: A functional gradient boosting approach", in *Conference on Artificial Intelligence in Medicine in Europe*, Springer, 2017, pp. 332–337.

[5] P. Deprez, P. V. Shevchenko, and M. V. Wüthrich, "Machine learning techniques for mortality modeling", *European Actuarial Journal*, vol. 7, no. 2, pp. 337–352, 2017.

[6] C. S. Kruse, B. Smith, H. Vanderlinden, and A. Nealand, "Security techniques for the electronic health records", *Journal of medical systems*, vol. 41, no. 8, pp. 1–9, 2017.

[7] K. El Emam, S. Rodgers, and B. Malin, "Anonymising and sharing individual patient data", *British Medical Journal*, vol. 350, 2015.

[8] European Parliament and Council, *Regulation 2016/679/EU on the protection of natural persons with regard to processing of personal data and on the free movement of such data*, 2016. [Online]. Available: `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN`.

[9] M. E. Kho, M. Duffett, D. J. Willison, D. J. Cook, and M. C. Brouwers, "Written informed consent and selection bias in observational studies using medical records: Systematic review", *British Medical Journal*, vol. 338, 2009.

[10] A. Wood, M. Altman, A. Bembenek, *et al.*, "Differential privacy: A primer for a non-technical audience", *Vand. J. Ent. & Tech. L.*, vol. 21, p. 209, 2018.

[11] A. Hundepool, J. Domingo-Ferrer, L. Franconi, *et al.*, *Statistical disclosure control*. Wiley New York, 2012, vol. 2.

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

[13] L. Sweeney, "K-anonymity: A model for protecting privacy", *International journal of uncertainty, fuzziness and knowledge-based systems*, vol. 10, no. 05, pp. 557–570, 2002.

[14] J. Bennett, S. Lanning, *et al.*, "The Netflix prize", in *Proceedings of KDD cup and workshop*, vol. 2007, 2007, p. 35.

[15] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets", in *2008 IEEE Symposium on Security and Privacy*, IEEE, 2008, pp. 111–125.

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

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

[18] D. B. Rubin, "Discussion: Statistical disclosure limitation", *Journal of official statistics*, vol. 9, no. 2, pp. 461–, 1993.

[19] C. M. de Melo, A. Torralba, L. Guibas, J. DiCarlo, R. Chellappa, and J. Hodgins, "Next-generation deep learning based on simulators and synthetic data", *Trends in cognitive sciences*, 2021.

[20] A. Koivu, M. Sairanen, A. Airola, and T. Pahikkala, "Synthetic minority oversampling of vital statistics data with generative adversarial networks", *Journal of the American Medical Informatics Association*, vol. 27, no. 11, pp. 1667–1674, 2020.

[21] T. E. Raghunathan, J. P. Reiter, and D. B. Rubin, "Multiple imputation for statistical disclosure limitation", *Journal of official statistics*, vol. 19, no. 1, p. 1, 2003.

[22] Finnish Social and Health Data Permit Authority Findata, *2021 presentation on Findata activities*, 2021. [Online]. Available: `https://findata.fi/wp-content/uploads/sites/3/2021/12/Findatan-infotilaisuus-8-12-2021-esitysmateriaalit.pdf`.

[23] I. J. Goodfellow, *NIPS 2016 tutorial: Generative adversarial networks*. [Online]. Available: `http://arxiv.org/abs/1701.00160`.

[24] S. Ulianova, *Cardiovascular disease dataset*, 2019. [Online]. Available: `https://www.kaggle.com/sulianova/`.

[25] B. Balle, G. Barthe, and M. Gaboardi, "Privacy amplification by subsampling: Tight analyses via couplings and divergences", *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[26] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs", *Advances in neural information processing systems*, vol. 30, 2017.

[27] L. Deng, "The MNIST database of handwritten digit images for machine learning research", *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[28] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms", *arXiv preprint arXiv:1708.07747*, 2017.

[29] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[31] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[32] O. Delalleau and Y. Bengio, "Shallow vs. deep sum-product networks", *Advances in neural information processing systems*, vol. 24, 2011.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[34] M. Abadi, A. Chu, I. Goodfellow, *et al.*, "Deep learning with differential privacy", in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[35] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?", in *2009 IEEE 12th international conference on computer vision*, IEEE, 2009, pp. 2146–2153.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets", *Advances in neural information processing systems*, vol. 27, 2014.

[37] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models", in *Proc. icml*, Citeseer, vol. 30, 2013, p. 3.

[38] D. P. Kingma, M. Welling, *et al.*, "An introduction to variational autoencoders", *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[39] G. E. Hinton, "Deep belief networks", *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[40] S. Kuhn, "Prisoner's Dilemma", in *The Stanford Encyclopedia of Philosophy*, 2019.

[41] L. J. Ratliff, S. A. Burden, and S. S. Sastry, "On the characterization of local Nash equilibria in continuous games", *IEEE transactions on automatic control*, vol. 61, no. 8, pp. 2301–2307, 2016.

[42] I. Dinur and K. Nissim, "Revealing information while preserving privacy", in *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2003, pp. 202–210.

[43] I. Mironov, "Rényi differential privacy", in *2017 IEEE 30th computer security foundations symposium (CSF)*, IEEE, 2017, pp. 263–275.

[44] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation", in *Annual international conference on the theory and applications of cryptographic techniques*, Springer, 2006, pp. 486–503.

[45] A. Rényi *et al.*, "On measures of entropy and information", in *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, Berkeley, California, USA, vol. 1, 1961.

[46] R. Bellman and R. Kalaba, "A mathematical theory of adaptive control processes", *Proceedings of the National Academy of Sciences*, vol. 45, no. 8, pp. 1288–1290, 1959.

[47] Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan, "Subsampled Rényi differential privacy and analytical moments accountant", in *The 22nd International Conference on Artificial Intelligence and Statistics*, PMLR, 2019, pp. 1226–1235.

[48] D. Chen, *GS-WGAN github-repository*, `https://github.com/DingfanChen/GS-WGAN`, 2020.

[49] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks", in *International conference on machine learning*, PMLR, 2017, pp. 214–223.

[50] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, *Differentially private generative adversarial network*, 2018.

[51] Y.-X. Wang and B. Balle, *Autodp: Automating differential privacy computation*, `https://github.com/yuxiangw/autodp`, 2013.

[52] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[54] A. Torfi, E. A. Fox, and C. K. Reddy, "Differentially private synthetic medical data generation using convolutional GANs", *Information Sciences*, vol. 586, pp. 485–500, 2022.

[55] The Finnish Medical Society Duodecim, *Current Care Guidelines: Treatment of hypertensive crisis*, Finnish, 2020. [Online]. Available: `https://www.kaypahoito.fi/hoi04010`.

[56] O. Troyanskaya, M. Cantor, G. Sherlock, *et al.*, "Missing value estimation methods for DNA microarrays", *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[58] B. K. Beaulieu-Jones, Z. S. Wu, C. Williams, *et al.*, "Privacy-preserving generative deep neural networks support clinical data sharing", *Circulation: Cardiovascular Quality and Outcomes*, vol. 12, no. 7, 2019.

[59] X. Zhang, S. Ji, and T. Wang, "Differentially private releasing via deep generative model (technical report)", *arXiv preprint : 1801.01594*, 2018.

[60] L. Frigerio, A. S. d. Oliveira, L. Gomez, and P. Duverger, "Differentially private generative adversarial networks for time series, continuous, and discrete open data", in *IFIP International Conference on ICT Systems Security and Privacy Protection*, Springer, 2019, pp. 151–164.

[61] J. Jordon, J. Yoon, and M. Van Der Schaar, "PATE-GAN: Generating synthetic data with differential privacy guarantees", in *International conference on learning representations*, 2018.

[62] Y. Long, S. Lin, Z. Yang, C. A. Gunter, and B. Li, "Scalable differentially private generative student model via PATE", *arXiv preprint : 1906.09338*, 2019.

[63] R. Torkzadehmahani, P. Kairouz, and B. Paten, "DP-CGAN: Differentially private synthetic data and label generation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[64] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with PATE", *arXiv preprint: 1802.08908*, 2018.

[65] J. Jordon, J. Yoon, and M. Van Der Schaar, "Pate-GAN: Generating synthetic data with differential privacy guarantees", in *International conference on learning representations*, 2018.

[66] R. McKenna, G. Miklau, and D. Sheldon, "Winning the NIST contest: A scalable and general approach to differentially private synthetic data", *arXiv preprint: 2108.04978*, 2021.

[67] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Privbayes: Private data release via Bayesian networks", *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 4, pp. 1–41, 2017.

[68] M. Walia, B. Tierney, and S. McKeever, "Synthesising tabular data using Wasserstein conditional gans with gradient penalty (WCGAN-GP).", in *AICS*, 2020, pp. 325–336.

[69] Y. Tao, R. McKenna, M. Hay, A. Machanavajjhala, and G. Miklau, *Benchmarking differentially private synthetic data generation algorithms*, 2022. arXiv preprint: 2112.09238.

[70] M. L. Fang, D. S. Dhami, and K. Kersting, "Dp-CTGAN: Differentially private medical data generation using CTGANs", in *AIME 2022, 20th International Conference on Artificial Intelligence in Medicine*, Springer, 2022.

[71] D. R. Cox, "The regression analysis of binary sequences", *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[72] T. K. Ho, "Random decision forests", in *Proceedings of 3rd international conference on document analysis and recognition*, IEEE, vol. 1, 1995, pp. 278–282.

[73] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance", in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 78.

[74] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms", *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[75] J. H. Zar, "Spearman rank correlation: Overview", *Wiley StatsRef: Statistics Reference Online*, 2014.

[76] V. M. Suriyakumar, N. Papernot, A. Goldenberg, and M. Ghassemi, "Chasing your long tails: Differentially private prediction in health care settings", in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021, pp. 723–734.

[77] F. Tramèr and D. Boneh, "Differentially private learning needs better features (or much more data)", in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. [Online]. Available: `https://openreview.net/forum?id=YTWGvpFOQD-`.