
KTH Stockholm

Department of Numerical Analysis

# Towards generating privacy-preserving and useful time series ECG data for arrhythmia detection

Master Thesis Report

Sijun John Tu

Supervisors: Anders Szepessy (KTH) and Shahid Raza (RISE)

**Abstract**

# STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

February 19, 2024

_____

Sijun John Tu

# Contents

# Notation

Mathematical conventions and notation used in this thesis:

| | |
|---|---|
| $\mathbb{R}$ | the real numbers |
| $\sqcup$ | disjoint set union |
| $\mathcal{N}(\mu, \sigma^2)$ | gaussian distribution with mean $\mu$ and variance $\sigma^2$ |

Additionally, we introduce the following conventions to describe various elements from different mathematical objects to make the notations and their meaning as consistent as possible:

| | |
|---|---|
| $\mathcal{S}$ | set of heartbeat samples |
| $s_i \in \mathbb{R}^L$ | seqeunce of ECG measurements |

# List of Figures

# 1 Introduction

## 1.1 Motivation

Data-driven technology and especially machine learning have gained a lot of momentum the past years. Models like ChatGPT or BERT heavily depend on large datasets that are available publicly. At the same time machine learning models are now being considered in other data sensitive domains like health care [see BK18; BND17; SSJ18; WS18]. One exciting field within health care is arrhythmia detection for heartbeats, where machine learning methods can aid physicians to detect irregular heartbeat conditions. Recently, several methods have been proposed, ranging from SVMs to neural networks [see review AIH18].

When working with those sensitive data, privacy plays a major role in general acceptance of those models. In some circumstances neural networks can memorise specific data samples, which constitutes a heavy privacy breach [see Fel21]. For example in [Car+18], Carlini et al. recovered credit card numbers from text completion models used by Google. Now governmental institutions like the European Union have established a right to privacy manifested in the General Data Protection Regulation laws[1]. Previous simple anonymisation attempts that simply removed some identifying attributes (e. g. name, birthday etc.) have been proven to be ineffective. For example, user profiles from the anonymised dataset used in the infamous Netflix prize have been reconstructed with the help of publicly available data from IMDB [NS08]. This is why technological advances in the area of privacy-preserving machine learning have increased in the past few years, with the development of various machine learning models that aim to preserve the privacy of individual data records. Protecting privacy becomes crucial for heartbeat data because it can be used to identify patients, thus heavily impacting the patient's privacy [see heartbeat biometrics Wan+18; Heg+11].

One promising solution [see Jor+22] is to replace the original, possibly sensitive data set with a synthetic data set that resembles the original raw data in some statistical properties. Much research has been done to generate tabular or image data, whereas dedicated time series data generation is still a *"burgeoning"* area of research according to a recent benchmark [Ang+23]. Regardless of the data type, data generators with no formal privacy guarantees have been shown to still be

[1] see https://gdpr-info.eu/

susceptible to privacy leaks [SOT22].
To improve privacy, this thesis aims to analyse the combination of synthetic data with tools from so-called differential privacy. Differential privacy has been developed by Dwork et al. [Dwo06] and is widely considered as the mathematical framework to rigorously provide privacy guarantees to privacy-preserving algorithms, relying on applied probability theory and statistics. This thesis will study existing architectures based on private generative AI models, as well as explore the possibility of new solutions. Experiments were conducted to assess the performance of these models using the MITBIH dataset on heartbeat arrhythmia [MM01]. Unfortunately, there is no free lunch and privacy always comes with a decrease in utility [SOT22]. A careful balance between privacy and utility needs to be established. However, we will challenge this trade-off and show that privacy and utility in the use case of anomaly detection can go hand in hand, because it can add some robustness to the model. This was first explored in [DJS19] for detecting anomalies.

## 1.2 Problem Definition

This thesis aims to examine how to generate privacy-preserving time series data for heartbeat arrhythmia detection. Let $\mathcal{S} = \{s_i\}_{i=1}^{N}$ denote a set of heartbeat samples, where $s_i = (s_i^0, ..., s_i^L)$ is a sequence of one-dimensional ECG measurements of fixed length $L$ corresponding to one heartbeat. Each heartbeat sequence is associated with a corresponding label denoting whether it is a normal or anomalous heartbeat according. Therefore we separate the set into normal heartbeats $\mathcal{N}$ and $\mathcal{A}$ (i. e. $\mathcal{S} = \mathcal{N} \sqcup \mathcal{A}$)

Firstly, we want to design a time series generator (TSG) that can model the true probability distribution $p(\mathcal{N})$ of the normal heartbeats. Here, we only consider normal heartbeats since for the subsequent task of arrhythmia detection we will follow an anomaly detection approach explained next. The aim of the TSG is to generate a synthetic data set $\widehat{\mathcal{N}}$ with distribution $p(\widehat{\mathcal{N}})$ that is "close" to the original data $p(\mathcal{N})$.
Secondly, the utility of the generated data is assessed in the downstream task of detecting anomalous heartbeats (heartbeat arrhythmia detection). We treat this task as an anomaly detection task based on reconstruction error [SWP22a, Section 3.2], i. e. we want to train a model only on normal heartbeats that can reconstruct those samples with low error, but give high reconstruction error when inputting an anomalous sample. Alternatively, one could treat this as a binary classification task,

that classifies a given heartbeat sample as either normal or anomalous. Since the ratio of those two classes are heavily imbalanced due to the nature of arrhythmias, we will favour the first approach.
Lastly, we will embed the generation procedure in a differential privacy setting. This will provide a theoretical framework to assess privacy.

## 1.3 Related Works and State of the Art

### Privacy in machine learning

A lot of past efforts have been put into improving the performance of machine learning methods, where the privacy aspect has been neglected. Due to the increased awareness about private individual data and policies like EU's GDPR laws, big tech companies like Apple, Google and even the US Census have been implementing privacy measurements in their data collection [see DKM19; Abo+19]. One of the first groundbreaking works on actually quantifying the privacy leakage in machine learning models has been studied in [Sho+17], where Shokri et. al. have designed a framework to perform membership inference attacks (MIA) on basic classification tasks. MIA on machine learning models try to infer whether a certain record has been used when training the respective model. This becomes a privacy issue when e. g. an adversary can infer whether a certain patient's data was used to train a model associated with a disease. Then the adversary can conclude that this particular patient likely has this disease [cf. Sho+17, p. 5]. Hence, their results indicate a strong vulnerability in terms of privacy for data-based models.
Several notions of privacy have been proposed in the last decade, among which Differential Privacy (DP) has emerged as the *"de-facto standard in data privacy"* [Kim+21]. Reasons for its popularity according to a recent survey [Gon+20] are among others:

1. DP is future-proof and requires no extra knowledge about the adversary.
2. DP provides rigorous privacy guarantees.
3. DP provides a notion of privacy budget, which can be adapted to the specific use case to balance privacy and utility.

We will revisit the definition and most important results in section 2 of this thesis. The basic idea is to add calibrated, random noise either to the data, during model training or to the output. Broadly speaking, differential private noise can be injected

in three different stages of the modelling pipeline: input, hidden or output layer [cf. ZCZ19b].

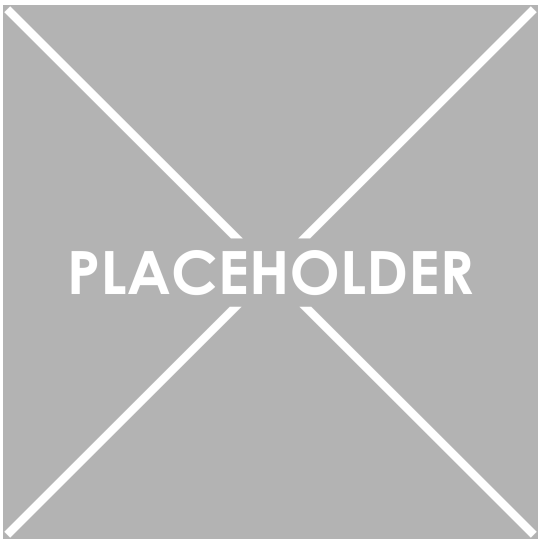


Figure 1: Different stages to add DP noise according to [Wan+23]

Applying some DP mechanism at the input stage can be seen as a preprocessing step to either hide sensitive attributes in the data or generating new synthetic dataset. Some earlier works include random perturbation methods described for instance in [Ma+23; FTS17]. According to [Wan+23] this approach is not utilised frequently because extra prior knowledge about the subsequent task is required to calibrate the right amount of noise. More recent methods focus exclusively on generating data samples by deep learning methods, that in turn employ a DP mechanism at gradient or output level.

Adding privacy in the hidden layer is sometimes referred to as gradient-level DP. Due to the iterative nature of most training algorithms, extra care needs to be taken to track the privacy loss caused by each iteration. Most notably there is a differential private version of stochastic gradient descent (SGD) called DP-SGD developed by Abadi et al [Aba+16] where the authors have designed a mechanism to track the privacy loss incurred while training. Differential privacy is achieved by clipping the gradient and then adding gaussian noise to the gradient. The clipping step is necessary to ensure that the gradient is bounded. Based on a more relaxed definition of DP, the authors in [Bu+20] propose an improved version of DP-SGD called NoisySGD.

At the output level there are several ways to implement DP. One highly cited approach called the “Functional Mechanism” followed by the authors in [Zha+12] perturbs the objective function, so it is independent of the number of training steps. A further refinement of this approach was researched in [Pha+17], where Phan et

al. developed an algorithm that puts adaptive noise on the features based on its contribution to the output.
Other interesting approaches to incorporating differential privacy into deep learning include the PATE learning method by Papernot et al. [Pap+17]. The idea behind this model is to train "teacher models" that will not be published, which in turn are used in a random manner to then train privacy-preserving "student" models.
In a distributed setting, approaches like federated learning [KMR15; MH19] have been proposed. Their privacy further analysed in [McM+18] for learning language models.

For a more in-depth review see e. g. [Ha+19; ZCZ19a; Wan+23]

### Data generation and Privacy

As we have mentioned earlier, ensuring privacy in machine learning applications is crucial when working with sensitive data. One might naively assume, that synthetic data without any formal privacy guarantees provides enough privacy already by design, but this unfortunately is not the case. Especially when generating with GAN-based networks, recent works have shown that although under some circumstances GANs can satisfy a weak notion of DP, but with a very high $\epsilon$ which corresponds to a very weak privacy guarantee [LSF21; SOT22; Jor+22]. Combining generative algorithms with DP however is a promising solution to mitigate the privacy issue [BDR19] which will be the focus of this thesis.
While we have outlined several techniques from the state of the art to ensure privacy, most of the methods are tailored for a specific model architecture or use case. On the other hand, synthetic data that has been generated with privacy guarantees can be used in any downstream task without privacy breach. To this end, several deep learning based architectures have been proposed. Following [Hu+23], one can broadly categorise them as follows:

- GAN-based
- Feature-based
- Autoencoder based [see e. g. Acs+17, for a generator based on a variational autoencoder that is trained with DP-SGD]
- Optimal transport based [see e. g. Cao+21, for generator based on the so-called Sinkhorn divergence]
- Stochastic simulation based [see e. g. Che+22, for a differentially-private diffusion model]

We will present the first two approaches in more detail in section 3.

#### Heartbeat arrhythmia detection

For the experiments conducted in this thesis we will use the common benchmark data set for heartbeat arrhythmia MIT-BIH [MM01]. It contains one-dimensional ECG measurements of 47 patients each lasting about 30 minutes. This kind of data is commonly referred to as time series data which due to its time dependency needs to be treated differently than tabular data. Each heartbeat is labelled as one of 16 heartbeat classes by experts. We will follow the Association for the Advancement of Medical Instrumentation's (AAMI) standard to divide those classes into normal and anomalous heartbeats [13]. Finding anomalous heartbeats, i. e. arrhythmia detection, can be linked to several common tasks found in machine learning. For example, one can view this problem as a binary classification problem, where one wishes to train a classifier, that given a heartbeat will classify this as either normal or anomalous. Since this requires a balanced dataset, we follow a different approach from anomaly detection. In particular, we will train a baseline model for arrhythmia detection based on so-called the reconstruction error [see SWP22b, for an in-depth survey on anomaly detection with times series]. This is a semi-supervised approach where a model is only trained on normal data. The model learns to encode and reconstruct normal data from a (lower-dimensional) latent space with low reconstruction error, whereas it will reconstruct anomalous data with a higher reconstruction error. This method will also be more useful in real-life applications since 1) heartbeat arrhythmias are rather scarce and 2) there is a lot of heartbeat data being collected but not labelled.
Some efforts have been taken to generate ECG data. Most of the recent approaches deploy a GAN based model to generate heartbeat data [see e. g. Zhu+19; DBW19; WGW20] getting favourable results. We will also follow a GAN based approach to generate synthetic ECG data. A very recent paper achieved even better result using a transformer architecture [see KD23].

## 1.4 Note on terminology

To avoid confusion about the different kinds of data sets used in this thesis, we establish some convention that will be followed throughout this work:

We refer to data set as being private, if this data set's privacy should be protected and therefore never be shared with the public. In contrast to that, a we refer to a

data set as being public, if we either do not aim to protect its privacy or this data set was generated with some rigorous privacy-preserving mechanisms.

Data that is coming from a real-life patients will be called the original data set. Data that is generated by some data-generating procedure is called synthetic data.

In the course of this thesis, we will look at different heartbeat samples that represent different heartbeat conditions. The heartbeat samples are classified according to those conditions. Following a guideline set by the AAMI we will simply label heartbeats as regular if they do not indicate any heartbeat disease or a non-symptomatic disease. Otherwise, those heartbeats will by labelled as anomalous heartbeats.

# 2 Theoretical background on Differential Privacy

In this chapter we briefly describe and derive the most important results from Cynthia Dwork's work on differential privacy that was first introduced in 2006 [Dwo06]. This summary heavily relies on her writings in [DKM19] and [DR+14] as well as lecture notes from [Gab16].

## 2.1 Defining differential privacy

Differential privacy (DP) should be understood as an agreement between the data holder and the data subject: the latter should not be "affected, adversely or otherwise, by allowing [her] data to be used in any study or analysis, no matter what other studies, data sets or information sources are available" [DR+14]. This addresses the paradox of learning something useful about a population while learning nothing about the individuals

**Example 2.1.1** (Randomised response)**.** In 1965 Warner [War65] proposes the following random answering procedure: In a study where participants are asked to answer with "Yes" or "No" whether they have engaged in an illegal or embarrassing activity $A$, they should:

1. Flip a coin
2. If the coin shows tails, then the participant should respond truthfully.
3. If the coin shows head, then the participant should flip the coin a second time and answer "Yes" if the second coin shows head and "no" otherwise.

This procedure ensures participants' privacy by "plausible deniability"; each participant's answer has non-zero probability of being truthful or not. By understanding the probabilities of the noise generation process, the data analyst can estimate the true number of "yes" and "no" answers. To this end, let $p$ be the true percentage of "yes" answers, $N$ the total number of participants, $n_{true}$ the true number of "yes" responses and $\hat{n}_{obs}$ the observed number of "yes" responses. We assume a fair coin with equal probability of showing heads or tails. Then the expected number of "yes" answers after applying the described procedure is:

$$\mathbb{E}(\text{"}Yes\text{"}) = \frac{1}{4}n_{true} + \frac{1}{4}(N - n_{true}) + \frac{1}{2}n_{true} = \frac{1}{4}N + \frac{n_{true}}{2} \tag{1}$$

We can estimate this using the $\hat{n}_{obs} \approx \mathbb{E}(\text{"}Yes\text{"}) = \frac{1}{4}N + \frac{n_{true}}{2}$ and finally solving for $n_{true}$ yields the estimate:

$$\hat{n_{true}} = 2\hat{n}_{obs} - \frac{1}{2}N \tag{2}$$

Now introduce some technical definitions, that will be need in order to define differential privacy.

**Definition 2.1.2** (Probability Simplex)**.** Given a discrete set $B$, the probability simplex over $B$ is defined as the set

$$\Delta(B) = \left\{ x \in \mathbb{R}^{|B|}, x_i \geq 0 \text{ and } \sum_i x_i = 1 \right\} \tag{3}$$

**Definition 2.1.3** (Randomised Algorithm)**.** A randomized algorithm $\mathcal{M}$ with domain $A$ and discrete range $B$ is associated with a mapping $M : A \to \Delta(B)$. On input $a \in A$ algorithm $\mathcal{M}$ outputs $\mathcal{M}(a) = b$ with probability $(M(a))_b$

**Definition 2.1.4** (Histogram representation of a data base)**.** Given a set $\mathcal{X}$, the universe of all possible records, the histrogram representation of a database $x$ is the vector

$$x \in \mathbb{N}^{|\mathcal{X}|} \tag{4}$$

in which each entry $x_i$ represents the number of elements in database $x$ of type $i \in \mathcal{X}$.

The previous definition of a database might sound cryptic at first, hence we will illustrate it with the following example:

**Example 2.1.5** (Database of patients)**.** Let $\mathcal{X} = \{P_1, ..., P_N\}$ be the set of $N$ distinct patients in a study. Then $x_1 = (1, 0, ..., 0) \in \mathbb{N}^N$ would correspond to patient $P_1$, $x_2 = (0, 1, 0..., 0) \in \mathbb{N}^N$ to patient $P_2$ and so on.

Equipped with this definition of a database one can now naturally define a way to measure "how much databases differ", i. e. in how many entries they differ.

**Definition 2.1.6** ($l_1$-norm of a database in histogram representation)**.** The $l_1$-norm of a database is a measure of the size of the database and defined as:

$$||x||_1 = \sum_{i=1}^{|\mathcal{X}|} |x_i| \tag{5}$$

This immediately gives rise to a notion of distance between two databases $x$ and $y$, namely:

$$||x - y||_1 = \sum_{i=1}^{|\mathcal{X}|} |x_i - y_i| \tag{6}$$

which basically counts the number of different entries.
Now we are ready to give the general definition of differential privacy:

**Definition 2.1.7** ($(\epsilon, \delta)$-DP)**.** A randomised algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is $(\epsilon, \delta)$-differentially private if for all outcomes $S \subset ran\mathcal{M}$ and for all databases $x, y \in \mathbb{N}^{|\mathcal{X}|}$, such that $||x - y||_1$ (i. e. they only differ in one element) we have

$$\mathbb{P}(\mathcal{M}(x) \in S) \leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{M}(y) \in S) + \delta \tag{7}$$

where the probability is taken over the randomness of $\mathcal{M}$. If $\delta = 0$, we say $\mathcal{M}$ is $\epsilon$-differentially private.

In other words, we call an algorithm $(\epsilon, \delta)$-DP if its outcome does not change "too much" on similar inputs. How much it is allowed to differ is specified by the factor $e^{\epsilon}$.

**Example 2.1.8** (Randomised response revisited)**.** We revisit the introductory example 2.1.1 and examine its privacy. It turns out that this simple procedure is differentially private! Without loss of generality let $x$ and $\hat{x}$ be two databases with $x_j = "Yes"$ and $\hat{x}_j = "No"$, $S = \{"Yes\}$. Then $\mathbb{P}(\mathcal{M}_{RR}(x_j) = "Yes") = \frac{3}{4}$, since the mechanism will return answer "Yes" (given the true answer is "Yes") if either the first coin toss is tails with probability $\frac{1}{2}$ or if the first and the second coin toss give heads with probability $\frac{1}{4}$. Similarly, we have $\mathbb{P}(\mathcal{M}_{RR}(\hat{x}_j) = "Yes") = \frac{3}{4}$ given that the true answer is "No" in this case. Thus we have:

$$\frac{\mathbb{P}(\mathcal{M}_{RR}(x_j) \in S)}{\mathbb{P}(\mathcal{M}_{RR}(\hat{x}_j) \in S)} = \frac{\mathbb{P}(\mathcal{M}_{RR}(x_j) = "Yes")}{\mathbb{P}(\mathcal{M}_{RR}(\hat{x}_j) = "Yes")} = \frac{\frac{3}{4}}{\frac{1}{4}} = 3 \tag{8}$$

The other cases follow analogously. Hence, this gives $(\ln 3, 0)-$ differential privacy.

## 2.2 Important results for Differential Privacy

The first question one might ask is whether adding randomness is crucial for differential privacy or whether there are alternative ways to ensure privacy without

adding randomness to it. This fundamental question is answered negatively in the following theorem:

**Theorem 2.2.1** (DP requires randomisation)**.** *Any non-trivial DP-mechanism requires randomisation.*

*Proof.* tba □

Hence, we established, that adding randomness is essential for differential privacy. But once we have obtained that level of privacy the output is immune to post-processing, e. g. for the case of privacy-preserving synthetic data once we have generated the data in a differential private setting, then the synthetic data will inherit the privacy guarantees for any subsequent task.

**Theorem 2.2.2** (Post-processing)**.** *Let $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R$ be a randomised algorithm that is $(\epsilon, \delta)$- DP. Further let $f : R \to R'$ an arbitrary function. Then $f \circ \mathcal{M}$ is also $(\epsilon, \delta)$ -DP.*

*Proof.* First fix data sets $x, y \in \mathbb{N}^{|\mathcal{X}|}$, s. t. $||x - y||_1 \leq 1$ and outcome $S' \subseteq R'$. Define a set $S = \{r \in R : f(r) \in S'\}$. Then we have:

$$\begin{aligned} \mathbb{P}(f(\mathcal{M}(x)) \in S') &= \mathbb{P}(\mathcal{M}(x) \in S) \\ &\leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{M}(y) \in S) + \delta \\ &= e^{\epsilon} \cdot \mathbb{P}(f(\mathcal{M}(y)) \in S') + \delta \end{aligned} \tag{9}$$

where the inequality follows from the $(\epsilon, \delta) - DP$ of $\mathcal{M}$. □

But if you employ two different DP mechnism at once, then their privacy degrades:

**Theorem 2.2.3** (Standard composition)**.** *Let $\mathcal{M}_1 : \mathbb{N}^{|\mathcal{X}|} \to R_1$ and $\mathcal{M}_2 : \mathbb{N}^{|\mathcal{X}|} \to R_2$ be two randomised algorithms that are $(\epsilon_1, \delta_1)$- and $(\epsilon_2, \delta_2)$ DP, then their composition defined by $\mathcal{M}_{12} : \mathbb{N}^{|\mathcal{X}|} \to R_1 \times R_2$, $\mathcal{M}_{12}(x) = (\mathcal{M}_1(x), \mathcal{M}_2(x))$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ DP.*

*Proof.* TBA □

**Theorem 2.2.4** (Group privacy)**.** *Let $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R$ be a randomised algorithm that is $(\epsilon, \delta)$- DP, then $\mathcal{M}$ is $(k\epsilon, ke^{k\epsilon}\delta)$- DP for groups of size $k$, i. e. it holds for databases $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $||x - y||_1 \leq k$ and for all $S \subseteq R$:*

$$\mathbb{P}(\mathcal{M}(x) \in S) \leq e^{k\epsilon} \cdot \mathbb{P}(\mathcal{M}(y) \in S) + k\delta \tag{10}$$

*Proof.* First fix data sets $x, y \in \mathbb{N}^{|\mathcal{X}|}$, s. t. $||x - y||_1 \leq k$ and outcome $S \subseteq R$. Now there exists a series of databases $z_0, ..., z_k$, such that $x = z_0$ and $y = z_k$ and $||z_{i+1} - z_i||_1 \leq 1$, i. e. we can find a series of databases that transforms $x$ into $y$ by removing or adding one record at a time. Then we have:

$$\begin{aligned}
\mathbb{P}(\mathcal{M}(x) \in S) &= \mathbb{P}(\mathcal{M}(z_0) \in S) \\
&\leq e^{\epsilon} \cdot \mathbb{P}(\mathcal{M}(z_1) \in S) + \delta \\
&\leq e^{\epsilon} \left(e^{\epsilon} \cdot \mathbb{P}(\mathcal{M}(z_2) \in S) + \delta\right) + \delta \\
&\leq ... \\
&= ke^{\epsilon} \cdot \mathbb{P}(f(\mathcal{M}(y)) \in S') + ke^{k\epsilon}\delta
\end{aligned} \tag{11}$$

□

## 2.3 Example of DP-mechanism: Gaussian Mechanism

Let $f : \mathbb{N}^{|\mathcal{X}|} \longrightarrow \mathbb{R}^d$ an arbitrary function mapping to a $d$-dimensional real space. $f$ can represent numerous models, e. g. a neural network, an SVM-classifier etc. We have seen from theorem 2.2.1 that in order to "privatise" the output of $f$, we need to add randomness to its output. One way to achieve this is to add gaussian noise, which is calibrated to mask the influence of a specific input. Because differential privacy aims to hide the influence of the input to the output, a natural quantity to consider when calibrating the noise is to look at how much $f$ will change, when using different inputs. This leads the following definition:

**Definition 2.3.1** ($l_2$-sensitivity)**.** Let $f : \mathbb{N}^{|\mathcal{X}|} \longrightarrow \mathbb{R}^d$ an arbitrary function, then its $l_2$-sensitivity is defined as:

$$\Delta f = \max_{\substack{x,y \in \mathbb{N}^{|\mathcal{X}|} \\ ||x-y||_1 \leq 1}} ||f(x) - f(y)||_2 \tag{12}$$

Now we can calibrate the noise according to its sensitivity which we can prove to satisfy differential privacy:

**Definition 2.3.2** (Gaussian Mechanism)**.** For a given function $f : \mathbb{N}^{|\mathcal{X}|} \longrightarrow \mathbb{R}^d$, privacy parameters $\epsilon \in (0, 1)$ and $\delta > 0$ define the gaussian mechanism $F(x)$ as

follows:

$$F(x) = f(x) + \mathcal{N}(0, \sigma^2) \tag{13}$$

where the variance is calibrated by the sensitivity of $f$ and the given privacy level, s. t. $\sigma \geq \frac{2\Delta f}{\epsilon} \ln(\frac{1.25}{\delta})$

**Theorem 2.3.3** (Gaussian Mechanism satisfies DP)**.** *The gaussian mechanism defined in definition 2.3.2 satisfies* $(\epsilon, \delta)$*-DP.*

The proof is rather lengthy and the curious reader is referred to read through [DR+14, Appendix A]

# 3 (Time Series) Data generation

This chapter gives a broad overview of different ML-based data generation algorithms. In particular, two algorithms will be explained in more detail that serve as a basis for the subsequent task of ECG time series data generation.

## 3.1 Overview

Time series data are sequences of data points in which there is a notion of time or ordering. Unlike tabular data, where each column corresponds to one feature, but it does not matter in which order one treats the different features. Time series are ubiquitous, common examples include weather data, financial transactions, energy consumption over time, stock prices etc.
We have chosen two architectures from the state of the art, that we will adapt to work on time series data. The first model is an example of a feature-based method, where a simple generative model is trained to map from a noise distribution to the data distribution. This is done by comparing the features of the synthetic data (or a suitable transformation thereof) with those of the original data. One particular instance of this class, DP-MERF [HAP20], has shown to give efficient and accurate results. Making this algorithm differentially private is straight-forward, since the loss function here can be separated into a term that is dependent on the original data and one that is not. So one only needs to introduce differential private noise to the data-dependent term once.
The second model follows a GAN-based approach. GANs introduced by Goodfellow et. al [Goo+14] have been studied extensively in recent works as they have shown promising results in the field of image generation. They consist of two networks, a generator and a discriminator, where those two networks play a zero-sum game: the generator aims to generate authentic data whereas the discriminator aims to distinguish between generated and real data.

## 3.2 DP-MERF

DP-MERF [HAP20] is an efficient all purpose data generation algorithm that is based on minimising the so-called Maximum Mean Discrepancy (MMD) between the real and the synthetic data distributions. It employs a so-called kernel mean embedding to transform the underlying probability distribution of the original data into a reproducing kernel hilbert space (RKHS). The distance between two distributions in the RKHS is then measured by the MMD. The authors mainly verified their

results using tabular data like the isolet dataset[1] but also image data, notably the MNIST [2] data set. It has not been used for time series data, but we will consider this data generation for generating time series data in this thesis, because according to a recent survey [Hu+23], DP-MERF delivers the best all purpose data generation performance.

### 3.2.1 Maximum Mean Discrepancy

There are different ways to measure the "distance" between two distributions $P$ and $Q$. On popular metric is the Maximum Mean Discrepancy (MMD) between $P$ and $Q$, where the random variables are projected into another feature space and the expected values are compared to each other in this space.

**Definition 3.2.1.1** (MMD)**.** Let $\phi : \mathcal{X} \to \mathcal{H}$, where $\mathcal{H}$ is a reproducing kernel hilbert space (RKHS) and $P$ and $Q$ some distributions over $\mathcal{X}$ and random variables $X \sim P$, $Y \sim Q$ given. Then the Maximum mean Discrepancy is defined as:

$$MMD(P, Q) = ||\mathbb{E}[\phi(X)] - \mathbb{E}[\phi(Y)]||_{\mathcal{H}} \tag{14}$$

Some "easy" features maps $\phi$ are for example:

**Example 3.2.1.2.** Let $P$ and $Q$ some distributions over $\mathcal{X}$ and random variables $X \sim P$, $Y \sim Q$ given.

- **Identity feature**: $\mathcal{X} = \mathcal{H} = \mathbb{R}^d$ and $\phi(x) = x$, then we have:

$$\begin{aligned} MMD(P, Q) &= ||\mathbb{E}[\phi(X)] - \mathbb{E}[\phi(Y)]||_{\mathcal{H}} \\ &= ||\mathbb{E}[X] - \mathbb{E}[Y]||_{\mathbb{R}^d} \end{aligned} \tag{15}$$

  So we only compare the two distributions in terms of their means.

[1] see https://archive.ics.uci.edu/dataset/54/isolet
[2] see http://yann.lecun.com/exdb/mnist/

- **Quadratic features**: $\mathcal{X} = \mathbb{R}$ $\mathcal{H} = \mathbb{R}^2$ and $\phi(x) = (x, x^2)$, then we have:

$$\begin{aligned} MMD(P,Q) &= ||\mathbb{E}[\phi(X)] - \mathbb{E}[\phi(Y)]||_{\mathcal{H}} \\ &= ||\mathbb{E}[(X, X^2)] - \mathbb{E}[(Y, Y^2)]||_{\mathcal{H}} \\ &= || \begin{pmatrix} \mathbb{E}[X] \\ \mathbb{E}[X^2] \end{pmatrix} - \begin{pmatrix} \mathbb{E}[Y] \\ \mathbb{E}[Y^2] \end{pmatrix} ||_{\mathbb{R}^2} \\ &= \sqrt{(\mathbb{E}[X] - \mathbb{E}[Y])^2 - (\mathbb{E}[X^2] - \mathbb{E}[Y^2])^2} \end{aligned} \tag{16}$$

  So here we compare the two distributions in terms of their means and their variance (or first and second moments respectively).

Now instead of computing a possibly high or even infinite dimensional transformation $\phi$ one can use the well-known kernel trick [SHS01]. Let $k(x, y) =< \phi(x), \phi(y) >_{\mathcal{H}}$ be a kernel with corresponding reproducing kernel hilbert space $\mathcal{H}$, then the computation of the MMD simplifies to:

$$\begin{aligned} MMD^2(P,Q) &= ||\mathbb{E}[\phi(X)] - \mathbb{E}[\phi(Y)]||^2_{\mathcal{H}} \\ &=< \mathbb{E}[\phi(X)], \mathbb{E}[\phi(X')] > - < \mathbb{E}[\phi(X)], \mathbb{E}[\phi(Y)] > - < \mathbb{E}[\phi(Y)], \mathbb{E}[\phi(X)] > \\ &\qquad + < \mathbb{E}[\phi(Y)], \mathbb{E}[\phi(Y')] > \\ &= \mathbb{E}[< \phi(X), \phi(X') >] - 2\mathbb{E}[< \phi(X), \phi(Y) >] + \mathbb{E}[< \phi(Y), \phi(Y') >] \\ &= \mathbb{E}[k(X, X')] - 2\mathbb{E}[k(X, Y)] + \mathbb{E}[k(Y, Y')] \end{aligned} \tag{17}$$

Where we introduced independent random variables $X, X' \sim P$, $Y, Y' \sim Q$.
In particular, this avoids computation in very high or even infinite dimensional spaces. A popular choice for the kernel is the

**Example 3.2.1.3** (Gaussian Kernel)**.** Let us define $k(x, y) = e^{\frac{||x-y||_2^2}{2}}$. This corresponds to an infinite-dimensional feature space, as can been seen with the following computation: tba

### 3.2.2 Random Fourier Features

Now given a training data set $X_m = \{x_i\}_{i=1}^m \sim P$ and a synthetic data set $X'_m = \{x_i\}_{i=1}^m \sim Q$ we can estimate their $MMD^2$ by estimating the expected value with a mean estimate:

$$\widehat{MMD}^2(X_m, X'_m) = \frac{1}{m^2}\sum_{i,j=1}^{m} k(x_i, x_j) + \frac{1}{m^2}\sum_{i,j=1}^{m} k(x'_i, x'_j) - \frac{2}{m^2}\sum_{i,j=1}^{m} k(x_i, x'_j) \quad (18)$$

Unfortunately, this will require $\mathcal{O}(m^2)$ computations which grows quadratically in the number of samples. This will be too big for a large training data set. As a remedy, the authors of [HAP20] propose to use Random Fourier Features based on a paper from 2007 [see RR07], to approximate the kernel $k$ using its fourier transform and Monte-Carlo-Simulation.

$$k(x, y) \approx \hat{\Phi}(x)^T \hat{\Phi}(x') \quad (19)$$

where $\hat{\Phi}(x) \in \mathbb{R}^D$ and $\hat{\Phi}_j(x) = \sqrt{\frac{2}{D}} cos(\omega_j^T x)$.
If we sample $w_j \sim \mathcal{N}$ from the Gaussian distribution, we are approximating the gaussian kernel.
Now we can approximate eq. (18) using those random fourier features:

$$\begin{aligned}\widehat{MMD}^2_{RFF}(X_m, X'_m) &\approx \frac{1}{m^2}\sum_{i,j=1}^{m} \hat{\Phi}(x_i)^T \hat{\Phi}(x'_j) + \frac{1}{m^2}\sum_{i,j=1}^{m} \hat{\Phi}(x_i)^T \hat{\Phi}(x'_j) - \frac{2}{m^2}\sum_{i,j=1}^{m} \hat{\Phi}(x_i)^T \hat{\Phi}(x'_j) \\ &= ||\frac{1}{m}\sum_{i=1}^{m} \hat{\Phi}(x_i) - \frac{1}{m}\sum_{j=1}^{m} \hat{\Phi}(x'_j)||^2_{\mathcal{H}} \quad (20)\end{aligned}$$

more stuff: https://gregorygundersen.com/blog/2019/12/23/random-fourier-features/

### 3.2.3 Vanilla DP-MERF

We can now introduce the version of DP-MERF presented in [HAP20]. Let $G_\theta$ denote a generative neural network with parameters $\theta$, i. e. given input $z \sim p(z)$ from some known probability distribution $p(z)$ we obtain a synthetic sample through $x' = G_\theta(z)$. We denote the distribution of the synthetic data samples by $Q$. Further,

let $X_m = \{x_i\}_{i=1}^m \sim P$ be our training data with true distribution $P$. By minimising

$$\begin{aligned}\widehat{\theta} &= \arg\min_\theta \widehat{MMD}^2_{RFF}(P,Q) \\ &\overset{eq.\ (20)}{=} \arg\min_\theta ||\frac{1}{m}\sum_{i=1}^m \hat{\Phi}(x_i) - \frac{1}{m}\sum_{j=1}^m \hat{\Phi}(x'_j)||_2^2 \\ &= \arg\min_\theta ||\hat{\mu}_P - \hat{\mu}_Q||_2^2 \end{aligned} \tag{21}$$

where we introduced notation $\hat{\mu}_P = \frac{1}{m}\sum_{i=1}^m \hat{\Phi}(x_i)$ and $\hat{\mu}_Q = \frac{1}{m}\sum_{i=1}^m \hat{\Phi}(x'_i)$. The DP version is obtained by observing that the original data set is entering the equation only through $\hat{mu}_P$ so we have to introduce noise only in this term by adding gaussian noise:

$$\tilde{\mu}_p = \hat{\mu}_P + \mathcal{N}(0, \sigma^2 I) \tag{22}$$

We choose $\sigma$ according to definition 2.3.2. For a given privacy level $(\epsilon, \delta)$ we need to compute the sensitivity $\Delta_{\hat{\mu}_P}$. There is an upper bound since we have by definition:

$$\begin{aligned}\Delta_{\hat{\mu}_P} &= \max_{\substack{X_m, X'_m \\ ||X_m - X'_m||_1 = 1}} ||\frac{1}{m}\sum_{i=1}^m \hat{\Phi}(x_i) - \frac{1}{m}\sum_{j=1}^m \hat{\Phi}(x'_j)||_2 \\ &= \frac{1}{m}\max_{x_m \neq x'_m} ||\hat{\Phi}(x_m) - \hat{\Phi}(x'_m)||_2 \\ &\overset{\Delta\neq}{\leq} \frac{1}{m}\max_{x_m \neq x'_m} ||\hat{\Phi}(x_m)||_2 + ||\hat{\Phi}(x'_m)||_2 \\ &\leq \frac{2}{m} \end{aligned} \tag{23}$$

where in the second equality we assumed without loss of generality that $X_m$ and $X'_m$ differ only in their last element, so that the other summands cancel each out and in the last inequality we are using the fact that $||\hat{\Phi}(\cdot)||_2 \leq 1$.

## 3.3 RTSGAN

### 3.3.1 Review: GANs

Generative adversarial networks (GAN) were first introduced in 2014 by Goodfellow et. al in [Goo+14] as an unsupervised learning algorithm for generative modelling. Since then it has been used extensively in image generation, where it excels at

generating high-resolution images. The original paper proposes a joint training of two machine learning models to output $\hat{p}_{model}$, usually neural networks, to implicitly model the unknown data distribution $p_{data}$ of a given training set.
Therefore, the first network denoted by $G$, commonly referred to as the generator, is able to sample from $\hat{p}_{model}$ by finding a mapping from some random noise $z$ and maps it to a sample $G(z;\theta_G)$ following $\hat{p}_{model}$. $G$ is parametrised by a set of weights $\theta_G$. The second model denoted by $D$, commonly referred to as the discriminator, aims to distinguish generated samples $\hat{x} = G(z,\theta_G)$ from real samples $x$, which can be treated as a binary classification model. The output $D(x;\theta_D)$ then is an estimate whether $x$ is a real sample, i. e. sampled from $p_{data}$ or fake, i. e. sampled from $\hat{p}_{model}$ respectively. Similarly, $D$ is parametrised by $\theta_D$.
During training the weights $\theta_D$ and $\theta_G$ are adjusted in order to minimise or maximise a certain loss:

- $D$ is trained to maximise the probability of correctly classifying real and generated samples.
- $G$ is trained to minimise the probability that $D$ identifies the generated samples.

This leads to the following minmax loss:

$$\min_G \max_D \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[1 - D(G(z))] \tag{24}$$

The training is done sequentially, i. e. in every epoch we first update the discriminator's weights using a some type of gradient descent that maximises eq. (24). Then the generator's weights are adjusted so that it minimises eq. (24). This optimisation can be regarded as a zero-sum game from game theory.
Although theoretical results for convergence where obtained in the original paper by Goodfellow et al., in practise GANs suffer from stability issues coming from exploding or vanishing gradients and mode-collapse [see Gui+20; JLO20, for in-depth review]. Thus, modifications to the loss function and training process that aim to stabilise training where developed, e. g. WGAN using the Wasserstein loss [ACB17].
In light of privacy concerns, standard GAN architectures without any formal privacy guarantees do not preserve any meaningful privacy of the training data. This negative results has been confirmed in [LSF21; SOT22]. Hence, a dedicated privacy mechanism has to be used. In particular, we will "privatise" the GAN architecture by using a DP-SGD when training the discriminator, similar to [Xie+18].

### 3.3.2 Time series Generation with RTSGAN

The authors in [Pei+21] propose a hybrid approach that employs a similar idea to our proposed AE-(DP)MERF algorithm; it combines an autoencoder architecture to learn a latent space and generates data within that space with a WGAN network.

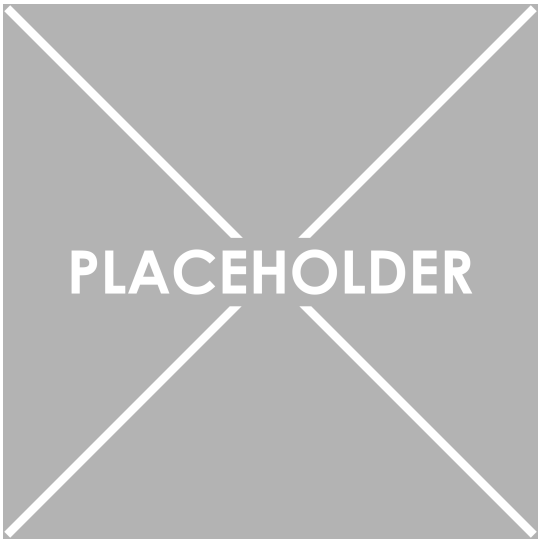


Figure 2: Architecture of RTSGAN from [Pei+21]

The authors also implement a mechanism to handle missing values, but we will not concern about this issue for the scope of this thesis.

- Autoencoder component: A gated recurrent unit (GRU) is used for both the encoder and the decoder. The encoder transforms the time series into a vector of fixed size in the latent space. This latent representation is then fed into the decoder which aims to reconstruct the time series from the latent space.

- Generator component The generator is based on the WGAN architecture. The main differences to a regular GAN lie in the loss function which is based on the so-called Wasserstein-1 distance and penalty term. Further, a Lipschitz-constraint is imposed on the discriminator through this penalty term.

$$\min_{G} \max_{D \in \mathcal{D}} \mathbb{E}_x[\log D(x)] - \mathbb{E}_z[D(G(z))] + \lambda \mathbb{E}_z[(||\nabla_z D(G(z))||_2 - 1)^2] \tag{25}$$

  where $\mathcal{D}$ denotes the set of 1-Lipschitz functions [1]). To impose this constraint, the authors chose to impose a penalty on the gradients as suggested in [ACB17].

[1] a differentiable function is 1-Lipschitz if and only if its derivative is bounded in norm by 1