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 23, 2024

_____

Sijun John Tu

ACKNOWLEDGEMENT

Thank you!

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 μ and variance σ^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].

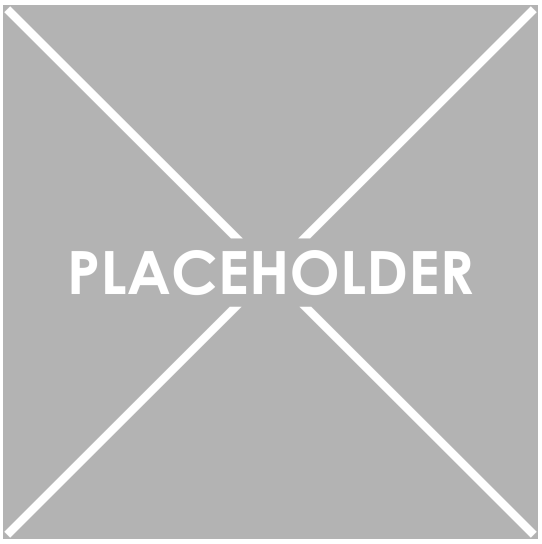

Figure 1: Different stages to add DP noise according to [Wan+23]

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 ϵ 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 ((ϵ, δ)-DP)**.** A randomised algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ϵ, δ)-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 ϵ-differentially private.

In other words, we call an algorithm (ϵ, δ)-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^{ϵ}.

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 (ϵ, δ)- DP. Further let $f : R \to R'$ an arbitrary function. Then $f \circ \mathcal{M}$ is also (ϵ, δ) -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 (ϵ_1, δ_1)- and (ϵ_2, δ_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 (ϵ, δ)- 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* (ϵ, δ)*-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 ϕ 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 ϕ 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 Equation (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_θ denote a generative neural network with parameters θ, 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

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

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 σ according to Definition 2.3.2. For a given privacy level (ϵ, δ) 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 θ_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 θ_D.
During training the weights θ_D and θ_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 Equation (24). Then the generator's weights are adjusted so that it minimises Equation (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.

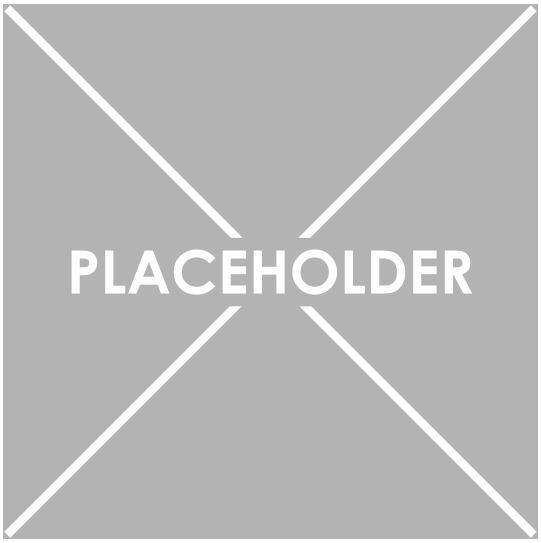

Figure 2: Architecture of RTSGAN from [Pei+21]

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

4 Models

In this chapter we will introduce the models that are being used for time series ECG data generation. In particular, we will modify the DP-MERF algorithm to work with time series data. Our modified version will be called AE-MERF for the non-privacy-preserving algorithm and AE-dpMERF for the DP version. To ensure comparability and privacy, we do some small modifications to the RTSGAN architecture as well and call the result AE-WGAN and AE-dpWGAN respectively.

4.1 AE-(dp)MERF

Out of the box, DP-MERF does not work well to generate time series data. We hypothesize, that the algorithm is not able to capture the temporal dependencies and inherent ordering in sequential data. The authors have verified their algorithm mainly on tabular data as well as image data, where it delivers competitive results. Hence, we want to leverage the capabilities and translate it into the sequential setting. Therefore, we use an autoencoder architecture, that maps the time series to a compact latent representation of fixed dimension. Data points in the latent space can then be treated as tabular data with no temporal dependencies. This approach is akin to [Hai+19] where Haidar et al. used an autoencoder with a GAN network to generate sequences of text. AE-(dp)MERF consists of two components.

- Autoencoder component: The autoencoder constists of an encoder and a decoder. The encoder will consists of two LSTM blocks to handle sequential data. It encodes the heart beat sequence as a vector of fixed length. The decoder will have the same architecture as the encoder but reversed. Thus it will learn to recover the heart beat sequence from the latent representation.

- DP-MERF component: The DP-MERF component consists of a simple feed forward neural network, called the generator, that takes in gaussian noise as input and maps it into the latent space. We compute the MMD distance between the original encoded data and the output of the generator and use this as the loss function. For the DP variant we add some noise to the RFF of the original data as described in Section 3.

We will proceed to train the two components separately, i. e. we will first train the autoencoder and then the generator component. The autoencoder will learn a latent representation of the original ECG time series. The DP-MERF generator is trained to generate data in the latent space which will then be transformed back to a ECG

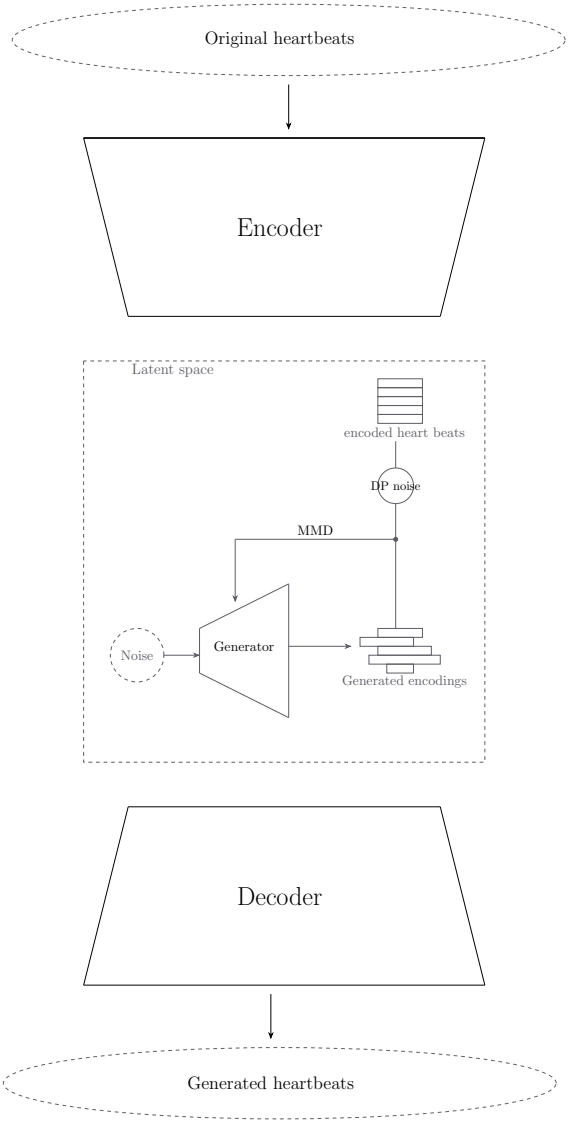

Figure 3: AE-(dp)MERF architecture

time series sample using the decoder. For DP version of this model just employs the same DP techniques from the original DP-MERF algorithm. The post-processing theorem for DP Theorem 2.2.2 ensures (differential) privacy for the generated ECG time series samples after being passed to the decoder.

4.2 AE-(dp)WGAN

We are making some small modification to the original architecture of RTSGAN. Firstly, we swap the GRU-based autoencoder with the same LSTM-based one from AE-(dp)MERF. This is to make sure, that both models are compared in the same latent space.
To implement a (differential) private GAN there are different ways:

- Input perturbation: One could add differential private noise to the input data

set before training. (Differential) Privacy is again preserved under the post-processing Theorem 2.2.2.

- Output perturbation: Similarly, one could add noise to the output of the model, once it is done training.
- Gradient perturbation: During training, adding a small amount of calibrated noise can also ensure (differential) privacy. This is implemented in the DP-SGD algorithm. The privacy level depends on the number iterations.

Although the latter approach deteriorates privacy at each training iteration, it is still the preferred way to "privatise" a GAN according to this paper by researcher from Facebook [MH20]. Hence, we will implement this in out GAN-based generator. Unfortunately, at the time of writing adding a gradient penalty is not supported [3] with the DP implementation of stochastic gradient descent in the `opacus` [4] package for `python`. Hence, instead of imposing the Lipschitz condition via the penalty term, we impose a strict constraint on the norm of the weights directly to enforce the Lipschitz condition. This approach was suggested initially by the authors of the original WGAN paper [ACB17]. We employ this in both the non-privacy preserving and DP version of this model to again ensure comparability.

[3] see github issue: https://github.com/pytorch/opacus/issues/31
[4] see https://opacus.ai/

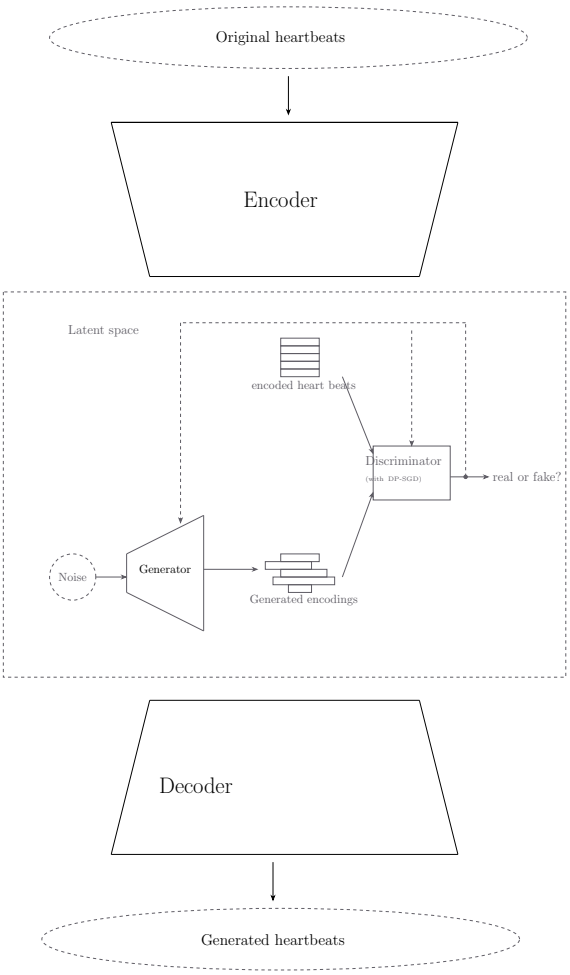

Figure 4: Architecture of AE-(dp)WGAN

5 Experiment

In this chapter we describe the experiments conducted on heartbeat data taken from the MIT-BIH Arrhythmia data set. Afterwards we present the results and draw insights. The aim is to assess the utility of synthetic data generated by the two algorithms mentioned in Section 4. The utility is measured in the downstream task of arrhythmia detection. We are testing differentially private and non-private versions of the models and compare them to a baseline model that is trained on the original data.

5.1 Data Set and Experiment Setup

The MIT-BIH is a commonly used benchmark data set for arrhythmia detection. It contains two channel ECG data collected in the 1980s from some 47 patients by the Arrhythmia Laboratory of Boston's Beth Israel Hospital (BIH; now the Beth Israel Deaconess Medical Center). 23 patients were chosen at random from a pool of over 4000 thousand patients, whereas the rest was chosen to include examples of clinically important but statistically uncommon arrhythmias. Hence, the proportion of arrhythmias in the whole data set is much larger than in reality. Studies suggest a percentage between 1.5% and 5% of patients with arrhythmias depending on the reference group [DH22].

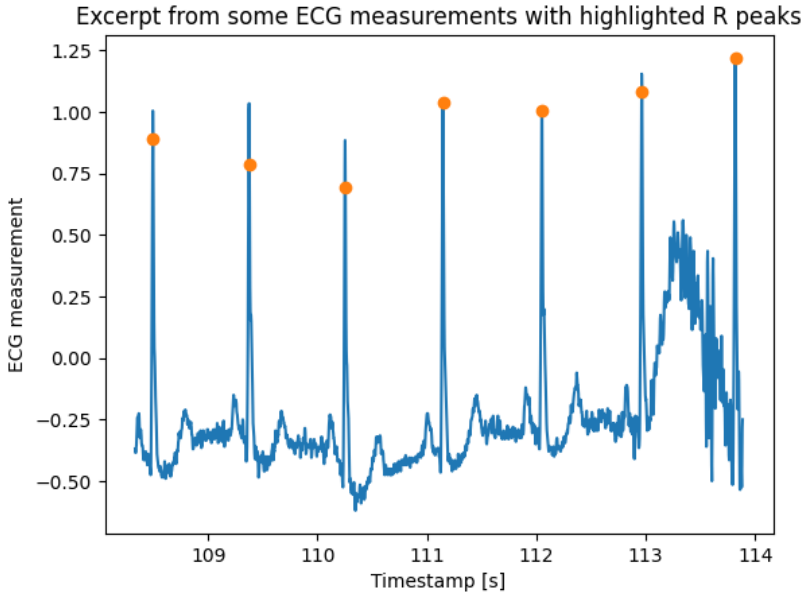

Figure 5: Excerpt from one ECG data sample, where the R peaks are highlighted

One single, regular heartbeat consists of one so-called QRS complex that consists of three blocks: a Q wave, R wave and S wave. Heartbeat arrhythmia is a medical

condition in which the patient experiences an abnormal rhythm of heartbeat. In most cases, this can be diagnosed by looking at the ECG: abnormal shapes in the different blocks or duration can indicate arrhythmia. This can be treated as an anomaly detection problem, where we train a model to learn the probability distribution of regular heartbeats (e. g. to capture their shape characteristics in terms of the building blocks). Based on some decision rule the model can detect samples that come from this distribution or were sampled from a different distribution.

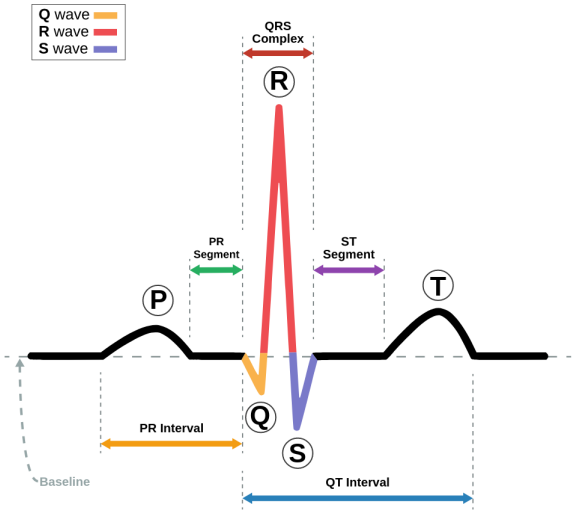

Figure 6: Schemativ represetation of a regular ECG wave, taken from wikipedia

Before we proceed with the training, we need to preprocess the data. Firstly, since the information contained in both channels are identical, one channel is discarded. We are roughly following the steps from ???REF. This involves normalising the data and extracting single heartbeats from the sequence. To this end, we employ the QRS detector that can detect the R peaks. This is already implemented in the `wfdb` [Xie+23] package for `python`. Then, from every peak we consider 90 time steps before and after that peak. This gives one heartbeat with sequence length 180. The associated beat type can then be extracted from the labelled data set. In total this gives over 100 000 single heartbeats each of length 180. We will refer to those samples as private data samples, of which we aim to protect the privacy. In contrast, the generated synthetic data samples will be called public data samples, that we can publish for public use with some associated privacy measure (the privacy is measured by the privacy budget given by the ϵ parameter from Definition 2.1.7). To ensure consistency of results and subsequent comparison of the different model, we split the original data into:

- a private training set that consists only of regular samples

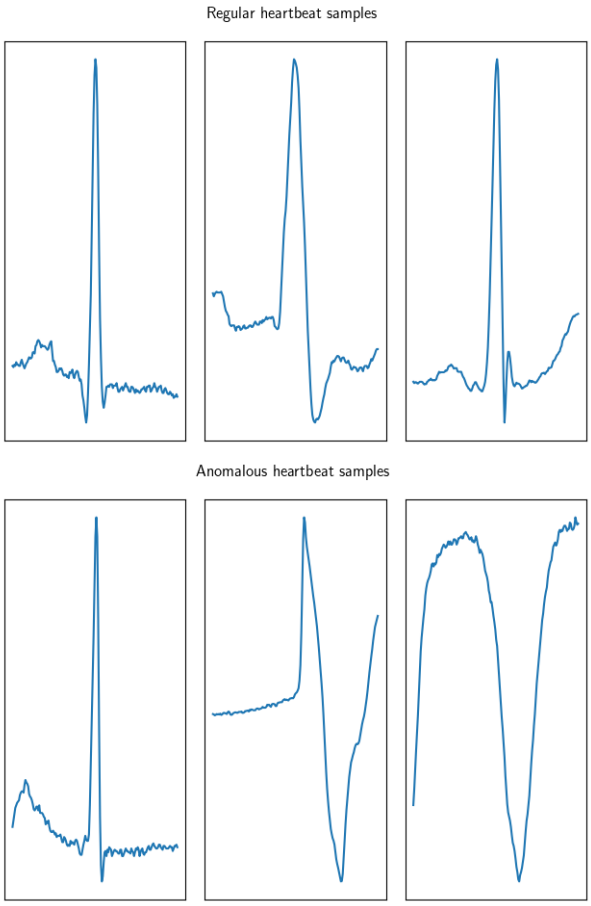

Figure 7: Regular and anomalous heart beat samples

- a private validation set that consists of roughly equal parts regular and anomalous samples
- a private test set that consists of roughly equal parts regular and anomalous samples

Therefore, we use the following proportions:

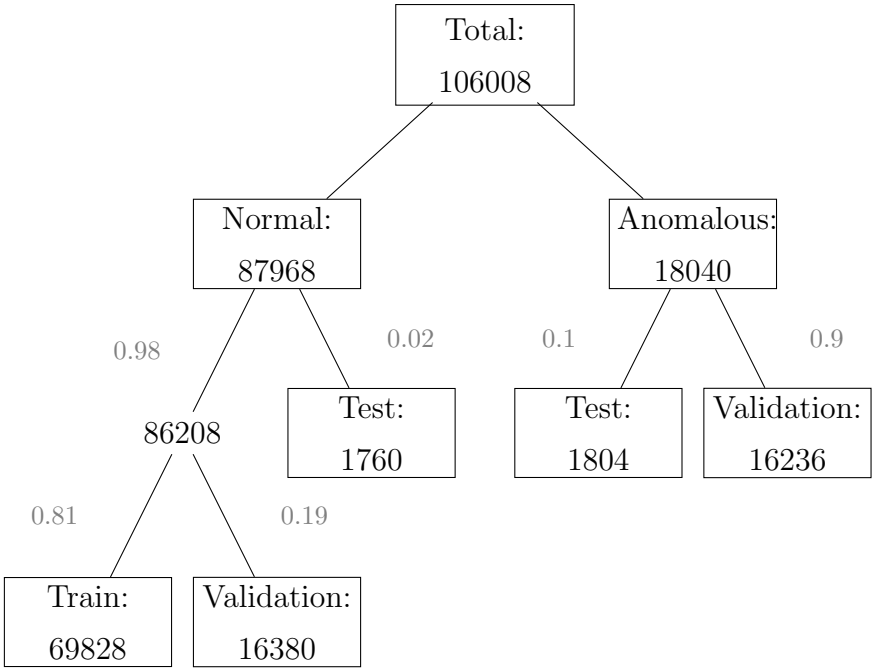

Figure 8: Division of the private data set into training, test and validation set

All preprocessing tasks and subsequent models are implemented in `python 3.10` on a machine running Arch Linux with kernel version 6.7.4 with 16 GB of RAM, an Nvidia RTX 3070 8GB, and AMD 7700 processor with 8 cores / 16 threads.

5.2 Baseline Model

We will establish a baseline model for arrhythmia detection based on an anomaly detection approach from machine learning. One could treat this also a binary classification problem, but due to the class imbalancy, this would decrease the performance. Hence we are following another idea: one popular solution is to train an autoencoder model on only the regular heartbeats. This way, the model is able to compress regular heartbeat sequence into a latent space and recover them with low reconstruction error. When seeing an anomalous sample, the model fails at properly reconstructing the sample, resulting in a high reconstruction error. We will adjust this error threshold using the validation set, that consists of roughly equal numbers of regular and anomalous samples.
The baseline model is an LSTM-autoencoder with two components:

- Encoder component: The encoder takes in the one-dimensional heartbeat sequence. There are two LSTM layers that encode the sequence into a latent space. We fixed the dimension of the latent space to 32.

- Decoder component: The decoder works similar to the encoder but the other

way around. Hence, we design it to take in an encoded vector representation of the heartbeat sequence in the latent space with dimension 32 and output a heartbeat sequence one time step after the other.

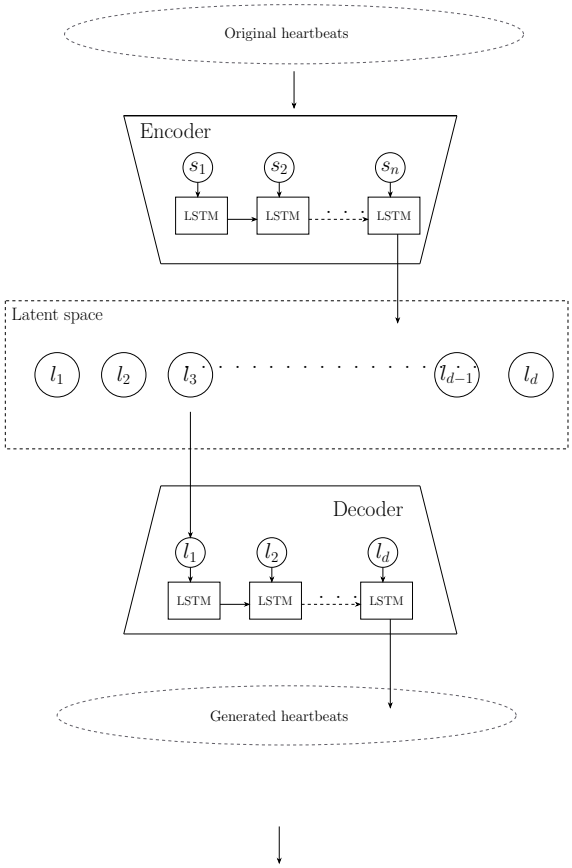

Figure 9: Architecture of baseline model

Training the Baseline Model

We train the model on the training set which consists of roughly 70 000 regular heartbeat samples. As loss function we take the mean-squared-error (MSE) between original and reconstructed sample. Training is done using the Adam optimiser with learning rate 5e-4, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for 20 epochs with batch size 1. As we can see from Figure 10, the loss function converges well indicating a (locally) optimal solution with an average reconstruction error slightly below 1. The loss for the validation set decreases similarly with the training error. This indicates that the model can generalise well to unseen regular heartbeat data.
When inputting a regular test sample, the model seems to capture all the visually important features and even removes some of the underlying noise.

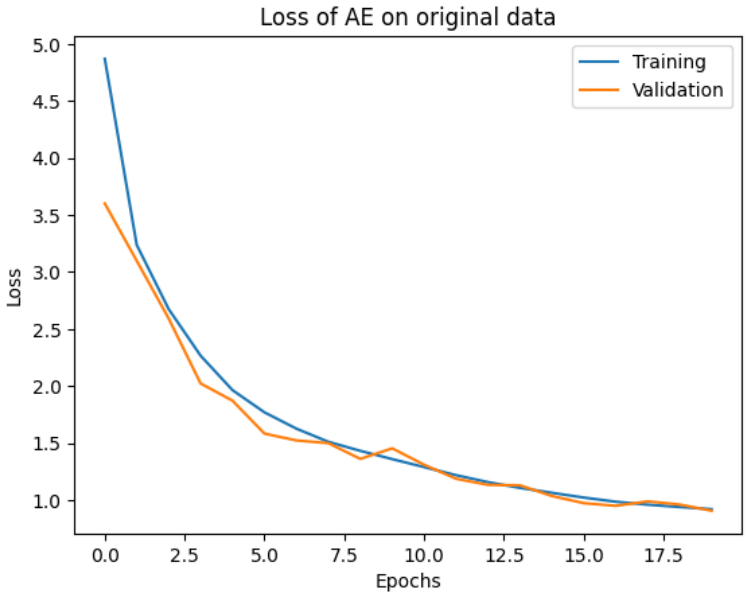

Figure 10: Loss over epoch for baseline model together with loss on validation set

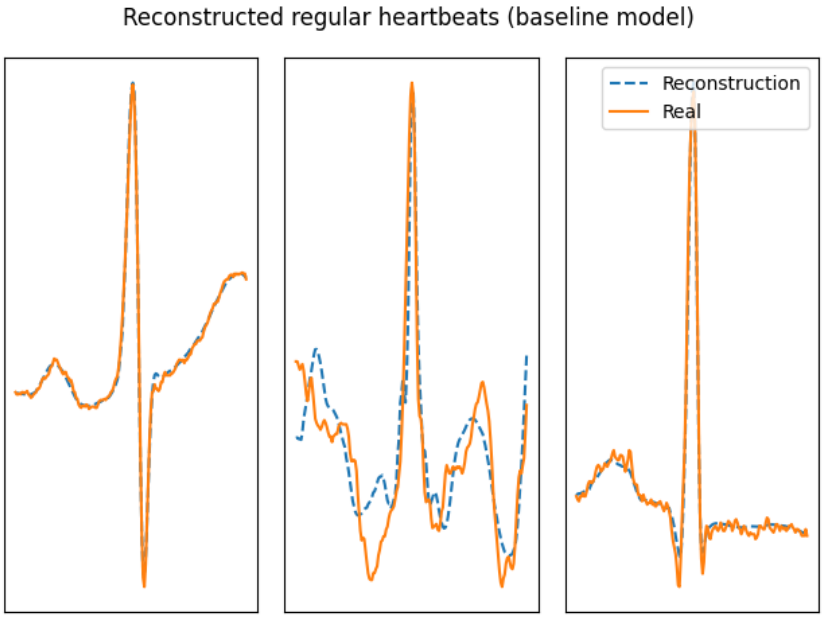

Figure 11: Regular test sample vs. reconstructed sample

Now we do the same but with an anomalous sample for demonstration purposes as can be seen in Figure 12.

Performance of Baseline Model

So visually, we can already verify that the baseline model seems to be able to distinguish between regular and anomalous samples. We now find an optimal threshold for the reconstruction error and measure the performance. Using the determined

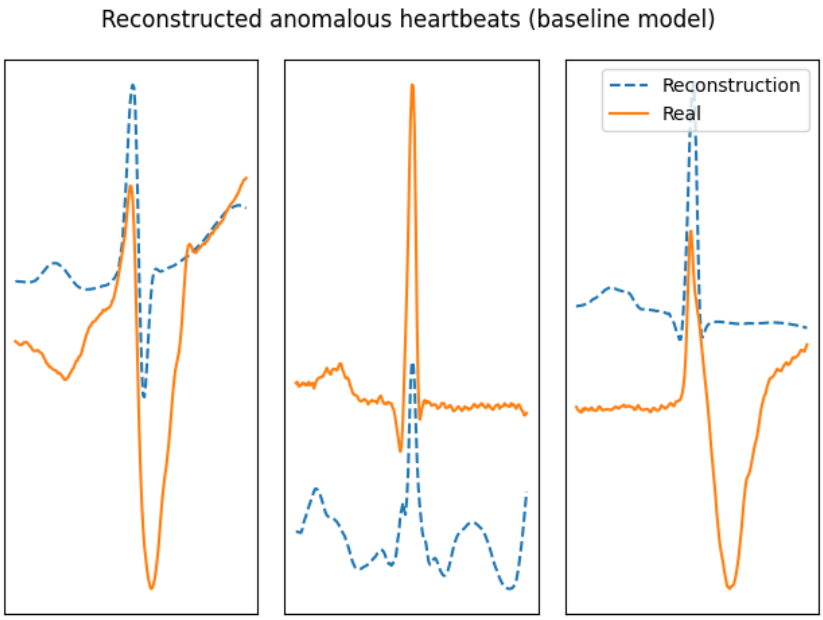

Figure 12: Anomalous test sample vs. reconstructed sample

theshold value, we do the classification as follows: we input a sample to the anomaly detection model, which outputs the reconstructed sample with some error. If the error is below the threshold, it will be classified as a regular heartbeat, otherwise it will be classified as an anomalous heartbeat. Firstly, let us plot the error distribution the held out validation set:

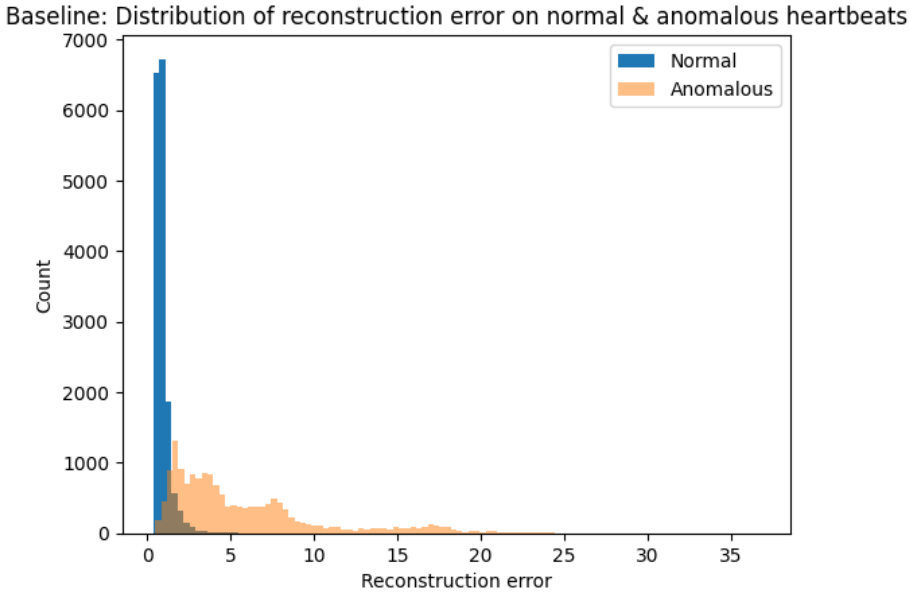

Figure 13: Distribution of reconstruction error for regular and anomalous samples in the validation set.

Clearly, the average error is much lower for regular heartbeats. Thus, the model is able to distinguish regular from anomalous samples by reconstruction error. We can

find the "good" threshold for the error by computing the percentage of correctly classified regular samples and correctly classified anomalous samples based on different threshold values. From Figure 14 we can see that the percentage of correctly classified regular samples slowly increases, while the percentage of correctly classfied anomalous samples decreases. This is to be expected, since a threshold value of 0 would simply classify all samples as anomalous, giving perfect classification for anomalous samples. Conversely, a very high threshold value (e. g. 20) would tend to classify each samples as regular, yielding perfect classification for regular samples. We need to strike a balance between those two extremes by examining Figure 14.

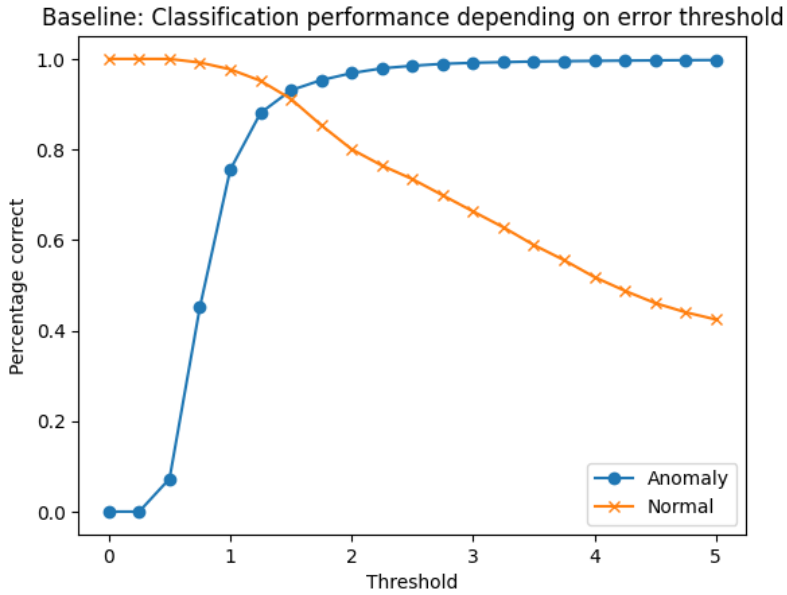

Figure 14: Percentages of correctly classified samples based on different threshold values

There are different ways to find a well-suited threshold depending on whether one want to have better performance on anomalous or regular samples. We choose to not favour any classification over the other, hence the neutral decision strategy for the threshold value is the one where both lines cross. Choosing the threshold here to be 1.5, we can compute different metrics that measure the performance of the arrhythmia detection (where TN=True Negatives, TP=True Positives, FN=False Negatives, FP=False Positives):

- Accuracy measures the overall percentage of correct classifications:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{26}$$

- Precision looks only on the samples that are labelled as anomalies and com-

putes the percentages of correctly detected anomalies:

$$Precision = \frac{TP}{TP + FP} \tag{27}$$

- Recall looks at all true anomalies and computes the percentage of correctly detected anomalies

$$Recall = \frac{TP}{TP + FN} \tag{28}$$

- F1 computes the an average of Precision and Recall

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{29}$$

Summary of Performance Evaluation

Since we will reuse the previous performance evaluation procedure for measuring the utility of the generated data, we quickly summarise the process:

1. Train an autoencoder on the synthetic training data set consisting of only regular samples. This synthetic training data is generated by our two different models.

2. Based on the distribution of the reconstruction errors on the private validation set for regular and anomalous samples, we determine a threshold value for classification.

3. We evaluate the utility of the generated data based on metrics for anomaly detection.

This approach commonly referred to as the "Train on synthetic, test on real" (TSTR) paradigm [EHR17] that is being used to measure utility of synthetic data.

5.3 Data Generation

Now we use the models presented in Section 4 to generate some synthetic heartbeat data. We will train the models on the private training data set, that consists only of regular samples. In turn, the models should be able to generate only regular heartbeat samples. Since both models are similar in architecture, the training procedure can be jointly described:

1. We train the autoencoder module separately from rather than jointly with the generator. The authors in [Pei+21] have found that there is no significant performance improvement when training jointly. This means, the models first learn how to encode the heartbeat time series in a shared latent space and how to decode it back from it. Conveniently, this is exactly the same autoencoder that is being used by the baseline model for arrhythmia detection, so we do not need to train it again.

2. We train the generator on encoded training data, i. e. we pass the private training data to the encoder which returns an encoded version of that training data in the latent representation. Then the generator trains to generate this latent representation.

3. After training, the generator generates a synthetic training set with the same number of samples as the private training set.

4. We evaluate the utility of the synthetic data by measuring its performance on anomaly detection following the procedure described in Section 5.2.

AE-MERF

We train the AE-MERF model with the following configuration:

Autoencoder	Embedding dimension	32
	Learning rate	5e-4
	Number of epochs	20
	Batch size	1
Generator (MERF)	Number of random features	2000
	Learning rate	1e-3
	Number of epochs	20
	Batch Size	7000

Table 1: Configuration for AE-MERF

We then let the model generate some training samples:
To assess the utility, we now train an anomaly detection model with a synthetic training data set. Therefore, we generate a synthetic data set with the same number of samples as the original private training data set, i. e. roughly 70 000 regular heart beat samples. We use the same architecture as for the baseline model - an LSTM autoencoder with embedding dimesion 32. As we can see from the loss in

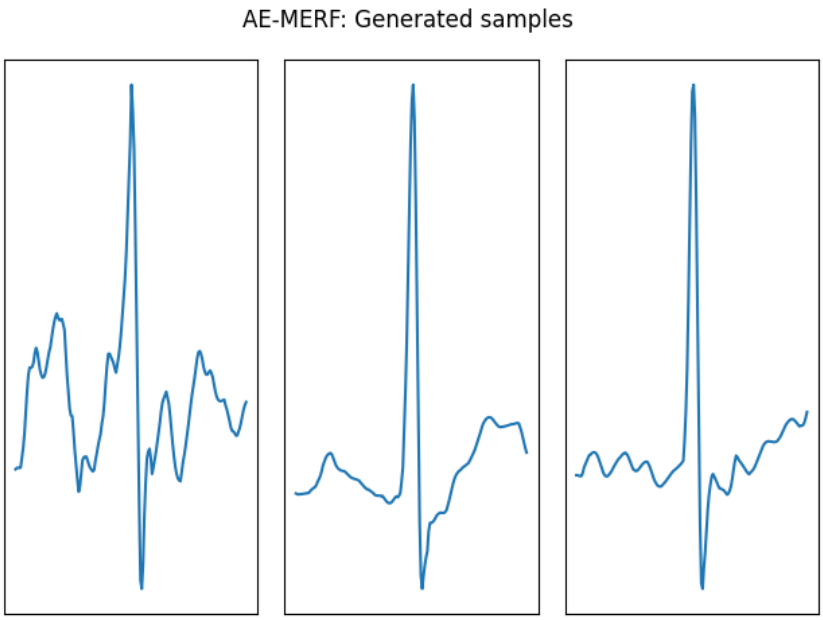

Figure 15: Generated regular heartbeats by AE-MERF model

Figure 16, the model converges on the synthetic training data, while still being able to generalise on the private validation set (that contains real-life samples). The error on the validation set is abit higher though in contrast to the baseline model. The training loss is again slightly below 1 while the validation loss averages around 2.

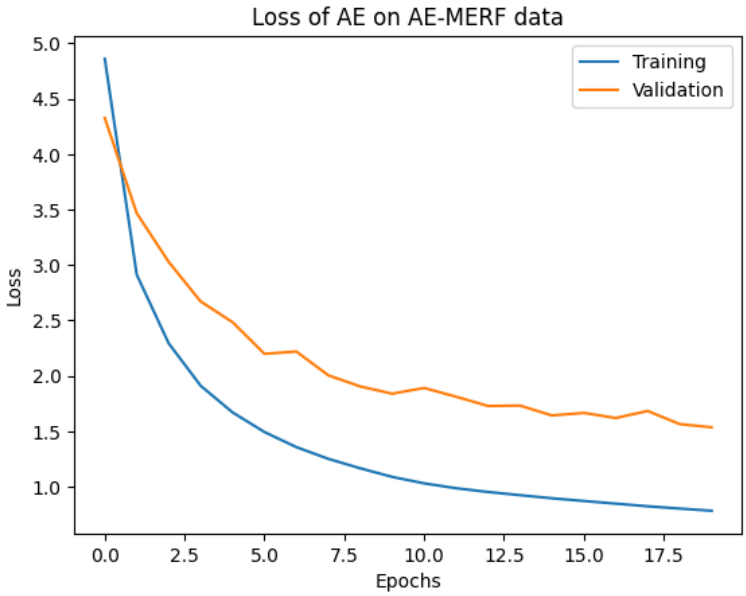

Figure 16: Loss over epoch for baseline model together with loss on validation set

Now following the established methodology, we compute a good threshold based on which we can distinguish between regular and anomalous samples:
This gives an error threshold of 1.75. This is slighty higher than threshold computed

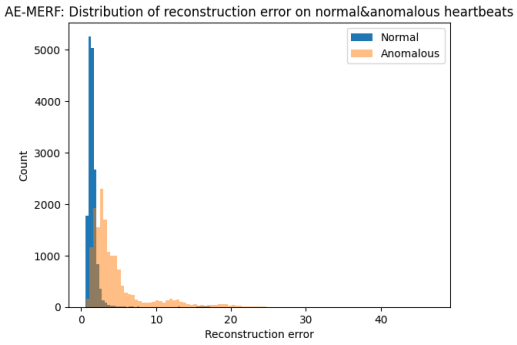

Figure 17: Caption

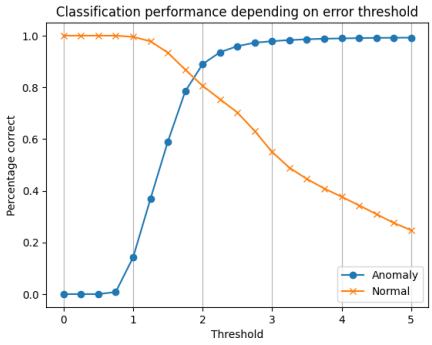

Figure 18: Caption

for the baseline model trained on private data, which indicates that the generator introduced some errors.

AE-WGAN

We train the AE-WGAN model with the following configuration:

Autoencoder	Embedding dimension	32
	Learning rate	5e-4
	Number of epochs	20
	Batch size	1
Generator (WGAN)	Discriminator steps	3
	Learning rate	9e-3
	Number of iterations	15000
	Batch Size	256

Table 2: Configuration for AE-WGAN

Again, we let the model generates some regular heartbeat samples:

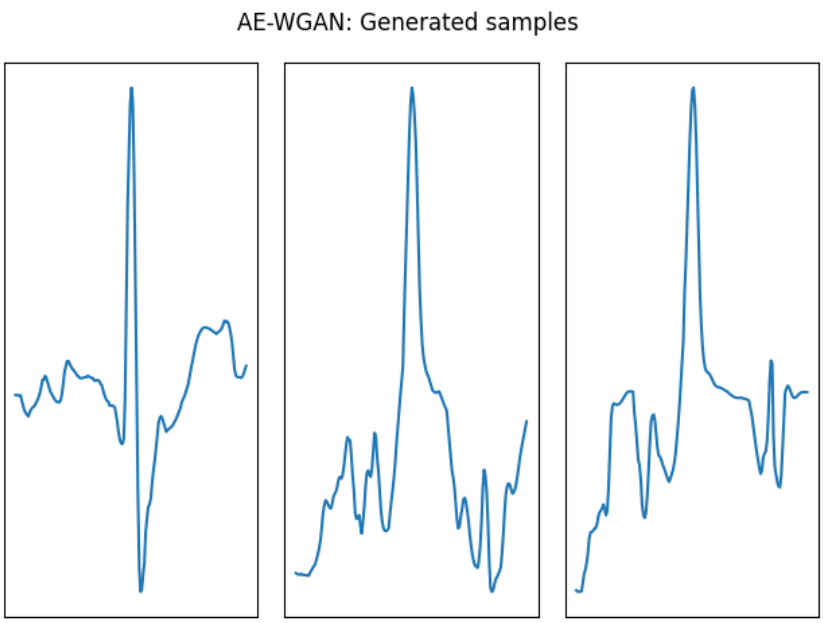

Figure 19: Generated regular heartbeats by AE-WGAN model

Compared to the generated samples from the AE-MERF model, the AE-WGAN model generates samples appear to be much noisier. We expect that the reconstruction error for the arrhythmia detection task will then also be much higher. We again train an LSTM-autoencoder on the synthetic training data generated by AE-WGAN. Looking at the loss, we can see that the validation error is much higher than the training error now. This leads to the assumption, that the generated samples differ quite much from the true samples, while still maintaining some important characteristics as the validation is also slowly decreasing.

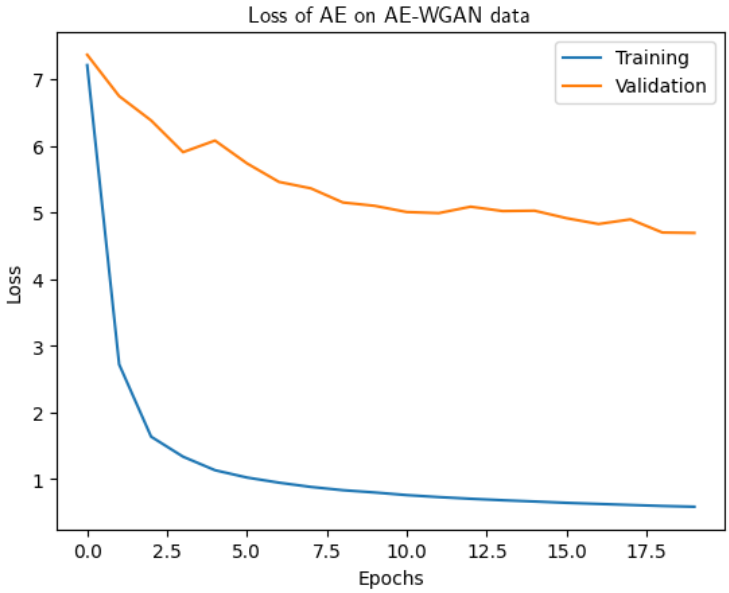

Figure 20: Loss over epoch for AE-WGAN model together with loss on validation set

This behaviour is mirrored when finding the threshold for anomaly detection:

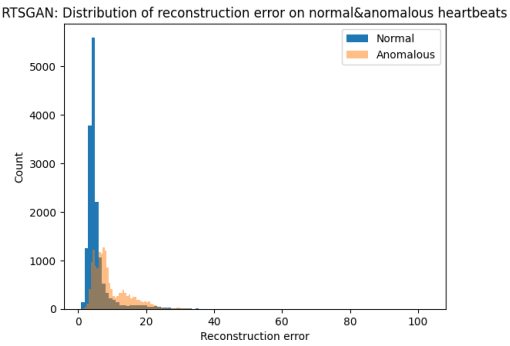

Figure 21: Caption

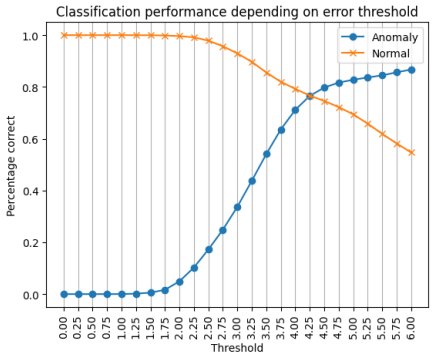

Figure 22: Caption

The threshold is much higher now, which again confirms the fact, that the samples generated by AE-WGAN deviate much more from the original samples.

5.4 Privacy-preserving Data Generation

We now follow the same procedure as previously but add DP noise to each model. How noise is calibrated and added is described in Section 4. We follow the same TSTR methodology as with the non-privacy-preserving models. For ease of reading, we omit the plots from training for now and refer to the appendix if needed.

AE-dpMERF

We use the DP version AE-dpMERF to generate some samples with DP guarantees. In particular, we generate samples with $\epsilon = 1, 0.5, 0.01, 0.001$ and assess their utility for anomaly detection.

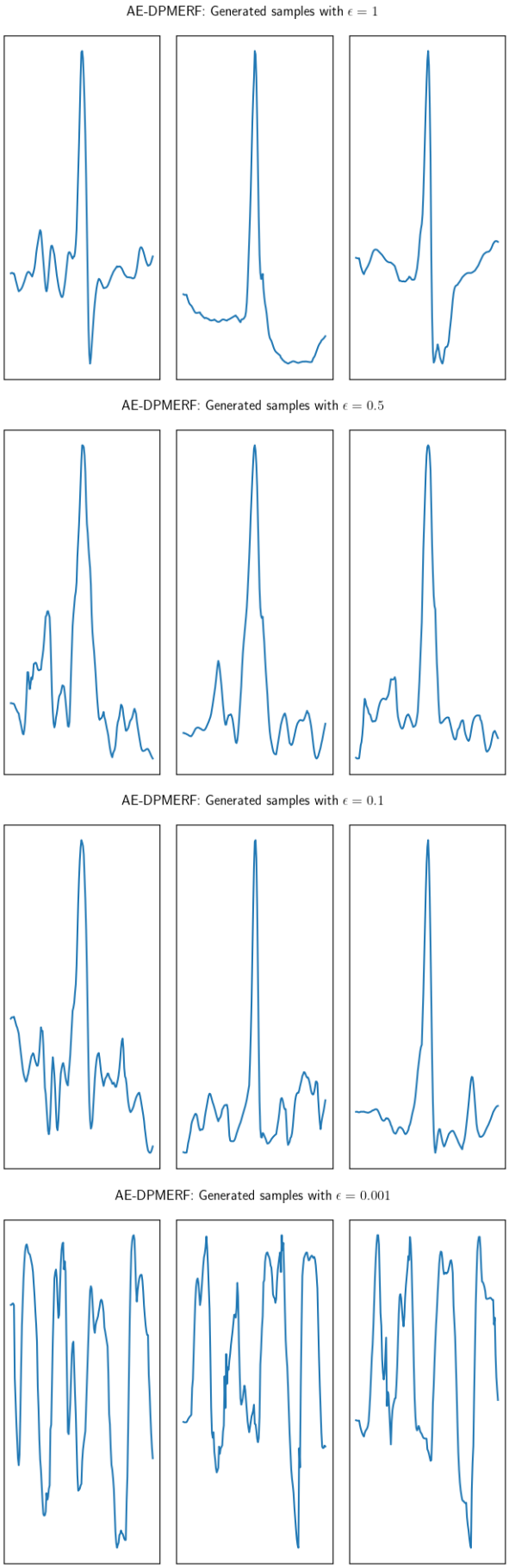

Figure 23: AE-dpMERF generated samples with different ϵ, the lower the value the higher the privacy guarantees

As we can see, the lower the ϵ privacy parameter the more noise is introduced. This gives stricter privacy guarantees, but for $\epsilon = 0.001$ too much noise is added and the generated samples visually do not appear to show a regular heartbeat. Clearly we can see from the generated samples in Figure 23, that more noise during training also results in more noise for the generated samples. In practice, it is recommended to use an ϵ value of 1, so AE-dpMERF with $\epsilon = 1$ already gives visually satisfying results [DKM19].

AE-dpWGAN

We use the DP version AE-dpWGAN to generate some samples with DP guarantees and $\epsilon = 35, 25, 5$. As the training for GANs is inherently very unstable and therefore sensitive to changes, we did not get satisfying results with even lower epsilon values.

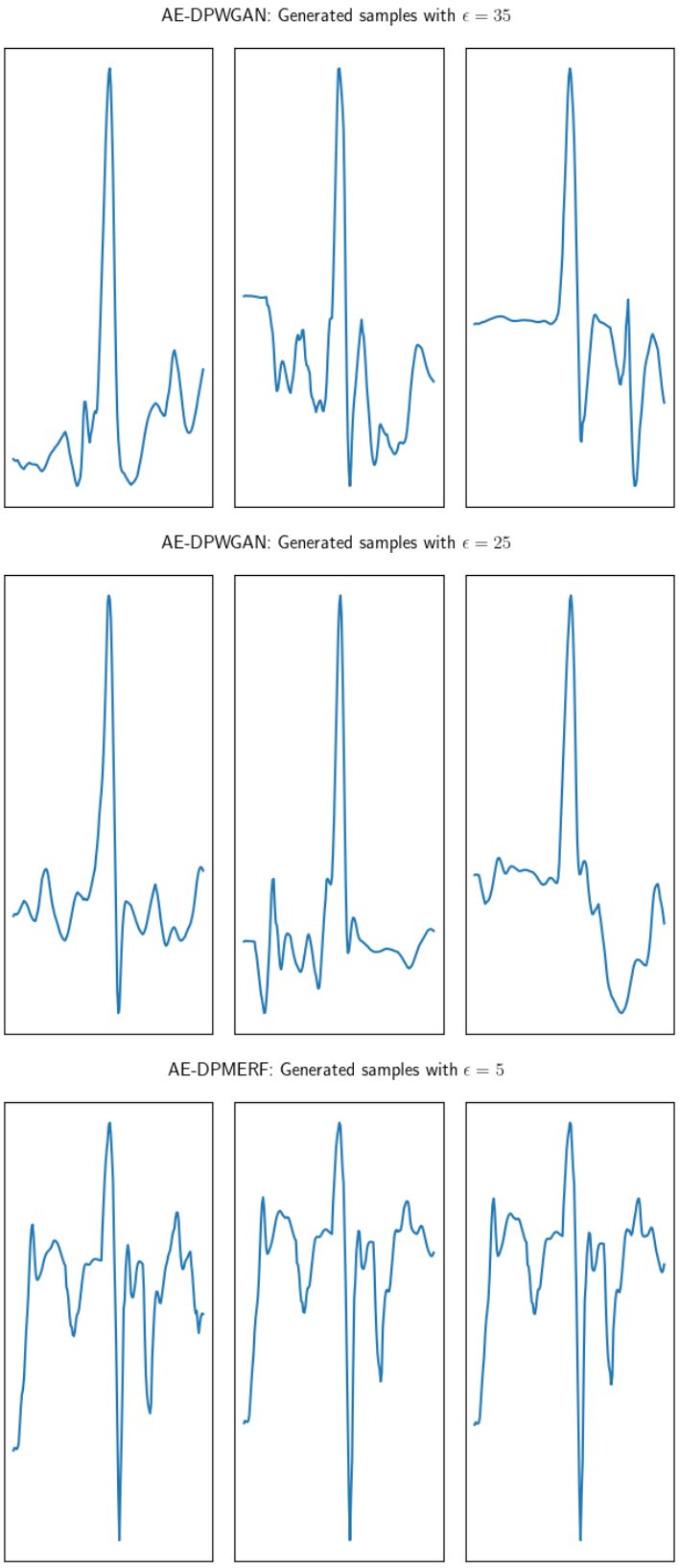

Figure 24: AE-dpWGAN generated samples with different ϵ, the lower the value the higher the privacy guarantees

As we can see, even with $\epsilon = 35$ there is a lot of visible noise in the generated

samples. This behaviour gets worse with decreasing ϵ. When training with $\epsilon = 5$ too much noise is added and the GAN gets really hard to train properly.

5.5 Results

We are now ready to compare all the different models in terms of their performance on anomaly detection. Table 3 shows a summary of all models in their performance scores. We also summarise this information in Figure 26. It is apparent that utility is lost already during (non-privacy-preserving) data generation. The loss in utility is around 10% for AE-MERF and 15% for AE-WGAN. Additionally, surprisingly we see that their is no significant loss in utility when introducing differential privacy to either models up to a certain ϵ value. For AE-dpMERF with $\epsilon = 0.5, 1$ the recall value is even higher than for the baseline model. Of course, the generated samples contain much more noise, thus leading to higher reconstruction error as we have seen before, but this does not impact the anomaly detection performance too much. Once the ϵ value is too low, thus too much noise is added during generation, the generated samples loose their utility for anomaly detection. For AE-dpMERF the lowest observed value is $\epsilon = 0.5$ for which we do not see a decrease in utility. On the other, synthetic samples generated by AE-dpWGAN become too noisy already with much higher ϵ values, e. g. $\epsilon = 5$.

Model	**Accuracy**	**Precision**	**Recall**	**F1**
Baseline	0.92	0.91	0.94	0.92
AE-MERF	0.83	0.85	0.79	0.82
AE-DPMERF ($\epsilon = 1$)	0.83	0.81	0.86	0.83
AE-DPMERF ($\epsilon = 0.5$)	0.81	0.81	0.80	0.80
AE-DPMERF ($\epsilon = 0.1$)	0.76	0.75	0.77	0.76
AE-DPMERF ($\epsilon = 0.01$)	0.48	0.47	0.41	0.43
AE-WGAN	0.76	0.75	0.75	0.75
AE-DPWGAN ($\epsilon = 35$)	0.74	0.74	0.75	0.74
AE-DPWGAN ($\epsilon = 25$)	0.72	0.73	0.71	0.71
AE-DPWGAN ($\epsilon = 5$)	0.56	0.56	0.57	0.56

Table 3: Summary of anomaly detection performance

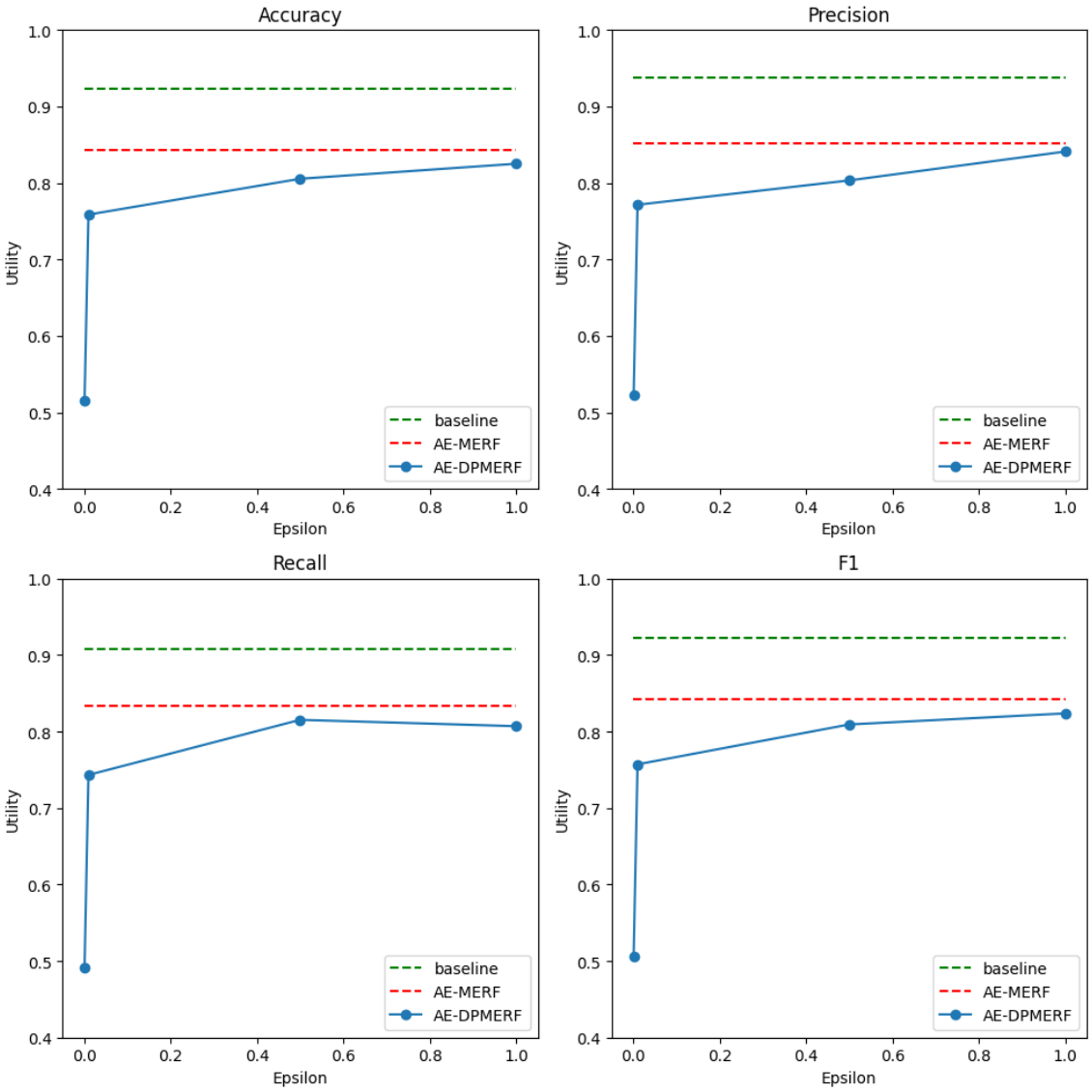

Figure 25: Utility loss over privacy budget ϵ for AE-(dp)MERF

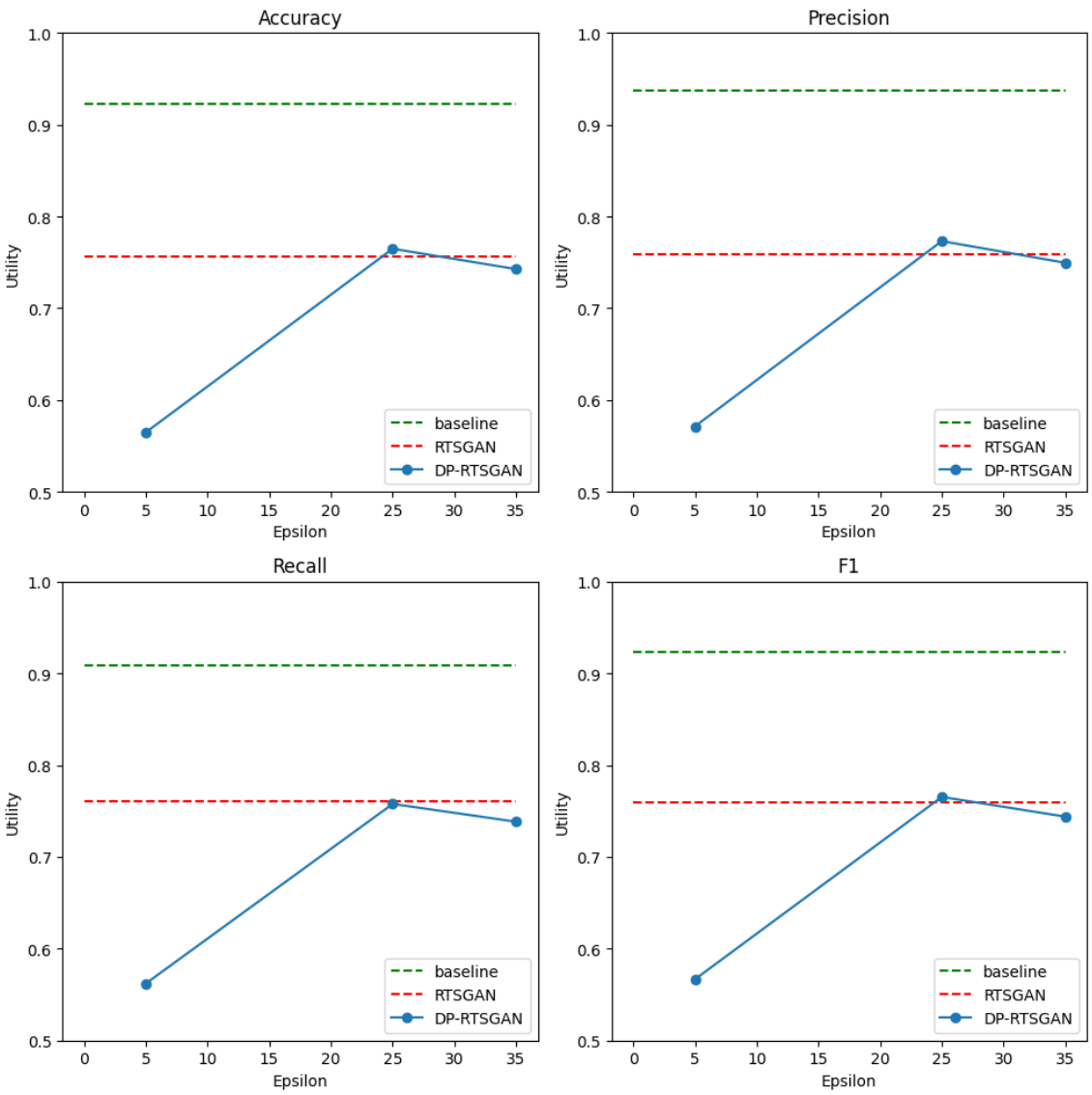

Figure 26: Utility loss over privacy budget ϵ for AE-(dp)WGAN

5.6 Contaminated Data Set

Now we contaminate the original training set that previously consisted of regular heartbeat samples only with some anomalous ones. This serves two purposes:

1. In real-life, labelled data is rare and heartbeat arrhythmia are scarce (1.5 - 5%). Having a contaminated training data set can simulate this scenario. Additionally, labelling errors or ambiguous cases can lead to a contaminated data set as well.

2. We can check how adding DP to the generation procedure will impact the robustness of the generated samples.

Therefore, we contaminate the training data with 1%, 2% and 5% of anomalous

heartbeat samples and repeat the data generation and anomaly detection as before only for the AE-(dp)MERF models with $\epsilon = 1, 0.5$ since they delivered promising results.

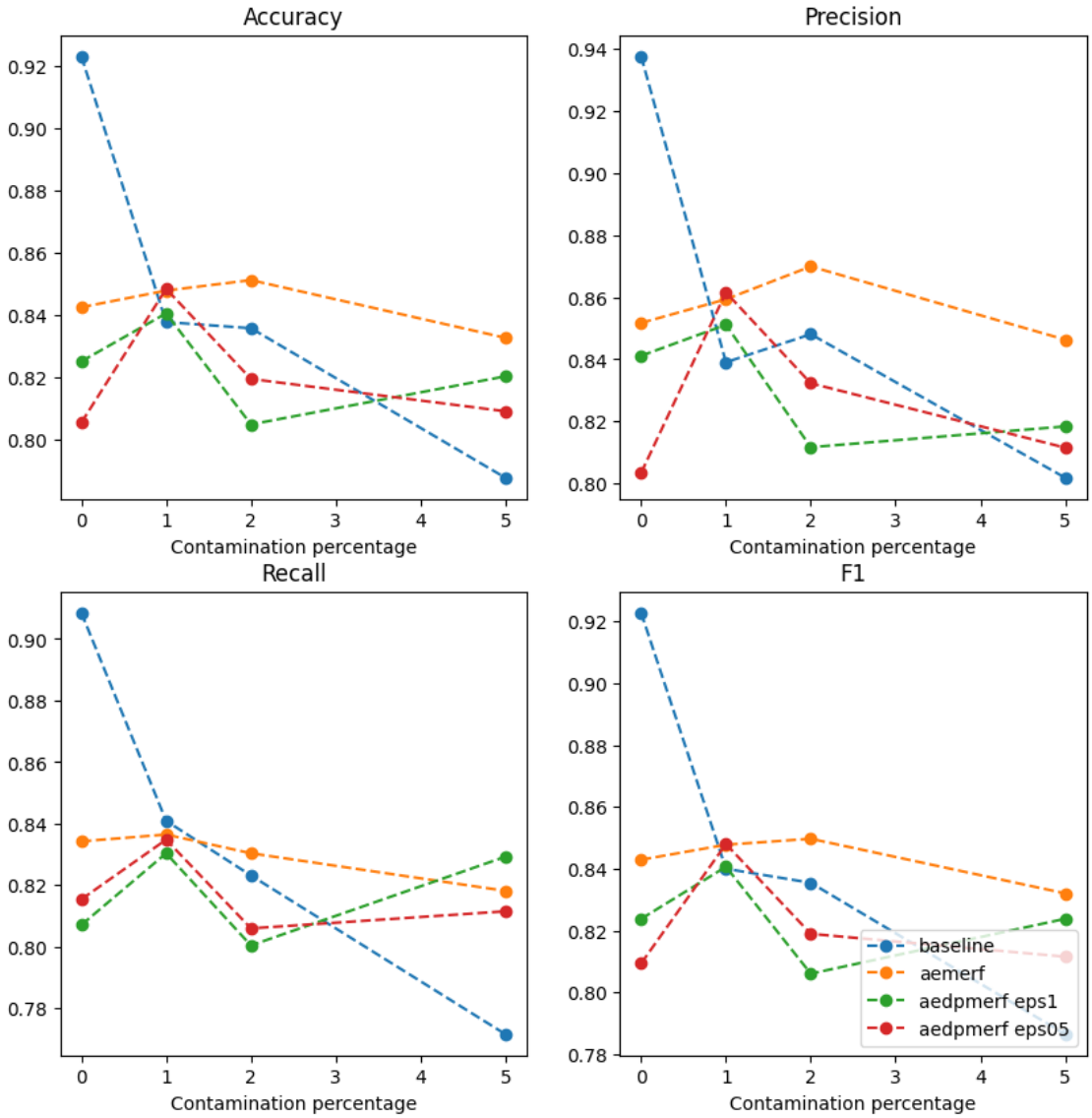

Figure 27: AE-(dp)MERF on contaminated training data

Looking only at the baseline model (blue line in Figure 27), we can clearly see that the performance decreases with higher contamination level. There is a slight increase for the Precision metric when going from 1% contamination to 2%. This is to be expected, since Precision measures the correctly detected anomalies among all detected anomalies, i.e. $Precision = \frac{TP}{TP+FP}$. When contaminating the training data set with anomalous samples, the number of undetected anomalies (FN) will increase and thus the number of wrongly detected anomalies (FP) will decrease, which in turn results in a higher Precision.

The non-privacy-preserving AE-MERF model behaves similarly, but the impact of

the contamination is smaller. This means that using AE-MERF generated samples is more robust than using the original private samples. This robustness transfers to the DP version: For both privacy-preserving models AE-dpMERF the performance metrics initially increase for a contamination level of 1%, before decreasing, when the contamination is increase to up to 5%. Surprisingly, for the highest contamination of 5%, the anomaly detection model trained with synthetic data outperforms the baseline model.

6 Discussion

6.1 (Privacy-Preserving) Data Generation

From our experiments, we can clearly see that the AE-(dp)MERF generated samples achieved a much higher utility compared to the GAN-based approach. This confirms the result from [Hu+23] stating DP-MERF is the "best all purpose generator". We found that AE-(dp)MERF also performs well on time series data, which has not been examined prior to this work to the best of our knowledge. Of course, further data and models need to be tested for verification. Adding privacy was straight forward to implement for AE-(dp)MERF and the training of the generator worked well in even lower ϵ regimes. This is not the case for the GAN-based approache: due to the inherent stability issues with training GANs, adding noise during the process imposes even more instabilities into the training process. Thus, training with even higher ϵ values required a careful selection of hyperparameters.

6.2 Utility-Privacy-Tradeoff

A lot if studies suggest, that adding (differential) privacy to a model decreases its utility. Our result suggest, that this tradeoff is more nuanced and depends on the use case. The conducted experiments show, that for reasonable ϵ values and depending on the generator, utility of synthetic, privacy-preserving data and privacy can go hand in hand. We have observed only a slight decrease in utility when employing DP to our models. For AE-(dp)MERF using $\epsilon =$ values as low as 0.5 did not "destroy" utility. The GAN-based model behaved similarly, but with much higher ϵ values ($\epsilon = 25$). However, this only applies to anomaly detection and does not say something about the quality of the generated samples. On the contrary, we have seen that the generated samples appear much noisier when we add differential privacy to the generation process.

6.3 Privacy and Robustness

In the last experiment, we contaminated the training set with anomalous samples.

7 Outro

7.1 Future Works

- test attacks to verify empirically
- test with other time series data
- rigorous hyperparameter tuning
-

7.2 Conclusion

Appendix

References

[Aba+16] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. "Deep Learning with Differential Privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS'16. ACM, Oct. 2016. URL: http://dx.doi.org/10.1145/2976749.2978318.

[Abo+19] Abowd, J., Ashmead, R., Simson, G., Kifer, D., Leclerc, P., Machanavajjhala, A., and Sexton, W. "Census topdown: Differentially private data, incremental schemas, and consistency with public knowledge". In: *US Census Bureau* (2019).

[Acs+17] Acs, G., Melis, L., Castelluccia, C., and De Cristofaro, E. "Differentially Private Mixture of Generative Neural Networks". In: *2017 IEEE International Conference on Data Mining (ICDM)*. 2017, pp. 715–720.

[Ang+23] Ang, Y., Huang, Q., Bao, Y., Tung, A. K. H., and Huang, Z. *TSGBench: Time Series Generation Benchmark*. 2023. arXiv: 2309.03755 [cs.LG].

[AIH18] Apandi, Z. F. M., Ikeura, R., and Hayakawa, S. "Arrhythmia Detection Using MIT-BIH Dataset: A Review". In: *2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA)*. 2018, pp. 1–5.

[ACB17] Arjovsky, M., Chintala, S., and Bottou, L. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[BK18] Beam, A. L. and Kohane, I. S. "Big Data and Machine Learning in Health Care". In: *JAMA* 319.13 (Apr. 2018), pp. 1317–1318. eprint: https://jamanetwork.com/journals/jama/articlepdf/2675024/jama_beam_2018_vp_170174.pdf. URL: https://doi.org/10.1001/jama.2017.18391.

[BDR19] Bellovin, S. M., Dutta, P. K., and Reitinger, N. "Privacy and synthetic datasets". In: *Stan. Tech. L. Rev.* 22 (2019), p. 1.

[BND17] Bhardwaj, R., Nambiar, A. R., and Dutta, D. "A Study of Machine Learning in Healthcare". In: *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 2. 2017, pp. 236–241.

[Bu+20] Bu, Z., Dong, J., Long, Q., and Su, W. J. "Deep learning with Gaussian differential privacy". In: *Harvard data science review* 2020.23 (2020), pp. 10–1162.

[Cao+21] Cao, T., Bie, A., Vahdat, A., Fidler, S., and Kreis, K. *Don't Generate Me: Training Differentially Private Generative Models with Sinkhorn Divergence*. 2021. arXiv: 2111.01177 [cs.LG].

[Car+18] Carlini, N., Liu, C., Kos, J., Erlingsson, Ú., and Song, D. "The Secret Sharer: Measuring Unintended Neural Network Memorization & Extracting Secrets". In: *CoRR* abs/1802.08232 (2018). arXiv: 1802.08232. URL: http://arxiv.org/abs/1802.08232.

[Che+22] Chen, J.-W., Yu, C.-M., Kao, C.-C., Pang, T.-W., and Lu, C.-S. "DP-GEN: Differentially Private Generative Energy-Guided Network for Natural Image Synthesis". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8377–8386.

[DBW19] Delaney, A. M., Brophy, E., and Ward, T. E. "Synthesis of Realistic ECG using Generative Adversarial Networks". In: *ArXiv* abs/1909.09150 (2019). URL: https://api.semanticscholar.org/CorpusID:202712887.

[DH22] Desai, D. S. and Hajouli, S. "Arrhythmias". In: *StatPearls [Internet]*. StatPearls Publishing, 2022.

[DJS19] Du, M., Jia, R., and Song, D. *Robust Anomaly Detection and Backdoor Attack Detection Via Differential Privacy*. 2019. arXiv: 1911.07116 [cs.LG].

[Dwo06] Dwork, C. "Differential privacy". In: *International colloquium on automata, languages, and programming*. Springer. 2006, pp. 1–12.

[DKM19] Dwork, C., Kohli, N., and Mulligan, D. "Differential privacy in practice: Expose your epsilons!" In: *Journal of Privacy and Confidentiality* 9.2 (2019).

[DR+14] Dwork, C., Roth, A., et al. "The algorithmic foundations of differential privacy". In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407.

[EHR17] Esteban, C., Hyland, S. L., and Rätsch, G. *Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs*. 2017. arXiv: 1706.02633 [stat.ML].

[Fel21] Feldman, V. *Does Learning Require Memorization? A Short Tale about a Long Tail*. 2021. arXiv: 1906.05271 [cs.LG].

[FTS17] Fukuchi, K., Tran, Q. K., and Sakuma, J. "Differentially Private Empirical Risk Minimization with Input Perturbation". In: *Discovery Science*. Ed. by Yamamoto, A., Kida, T., Uno, T., and Kuboyama, T. Cham: Springer International Publishing, 2017, pp. 82–90.

[Gab16] Gaboardi, M. *CSE711: Topics in Differential Privacy*. Lecture Notes. 2016.

[Gon+20] Gong, M., Xie, Y., Pan, K., Feng, K., and Qin, A. "A Survey on Differentially Private Machine Learning [Review Article]". In: *IEEE Computational Intelligence Magazine* 15.2 (2020), pp. 49–64.

[Goo+14] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Vol. 27. Curran Associates, Inc., 2014.

[Gui+20] Gui, J., Sun, Z., Wen, Y., Tao, D., and Ye, J. *A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications*. 2020. arXiv: 2001.06937 [cs.LG].

[Ha+19] Ha, T., Dang, T. K., Dang, T. T., Truong, T. A., and Nguyen, M. T. "Differential Privacy in Deep Learning: An Overview". In: *2019 International Conference on Advanced Computing and Applications (ACOMP)*. 2019, pp. 97–102.

[Hai+19] Haidar, M. A., Rezagholizadeh, M., Omri, A., and Rashid, A. "Latent Code and Text-based Generative Adversarial Networks for Soft-text Generation". In: Jan. 2019, pp. 2248–2258.

[HAP20] Harder, F., Adamczewski, K., and Park, M. "Differentially Private Mean Embeddings with Random Features (DP-MERF) for Simple & Practical Synthetic Data Generation". In: *CoRR* abs/2002.11603 (2020). arXiv: 2002.11603. URL: https://arxiv.org/abs/2002.11603.

[Heg+11] Hegde, C., Prabhu, H. R., Sagar, D., Shenoy, P. D., Venugopal, K., and Patnaik, L. M. "Heartbeat biometrics for human authentication". In: *Signal, Image and Video Processing* 5 (2011), pp. 485–493.

[Hu+23] Hu, Y., Wu, F., Li, Q., Long, Y., Garrido, G. M., Ge, C., Ding, B., Forsyth, D., Li, B., and Song, D. *SoK: Privacy-Preserving Data Synthesis*. 2023. arXiv: 2307.02106 [cs.CR].

[JLO20] Jabbar, A., Li, X., and Omar, B. *A Survey on Generative Adversarial Networks: Variants, Applications, and Training*. 2020. arXiv: 2006.05132 [cs.CV].

[Jor+22] Jordon, J., Szpruch, L., Houssiau, F., Bottarelli, M., Cherubin, G., Maple, C., Cohen, S. N., and Weller, A. *Synthetic Data – what, why and how?* 2022. arXiv: 2205.03257 [cs.LG].

[KD23] Kaleli, H. S. and Dehalwar, V. "Generation of Synthetic ECG Signal Using Generative Adversarial Network With Transformers". In: *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (2023), pp. 1–6. URL: https://api.semanticscholar.org/CorpusID:265406122.

[Kim+21] Kim, J. W., Edemacu, K., Kim, J. S., Chung, Y. D., and Jang, B. "A survey of differential privacy-based techniques and their applicability to location-based services". In: *Computers & Security* 111 (2021), p. 102464.

[KMR15] Konečný, J., McMahan, B., and Ramage, D. *Federated Optimization:Distributed Optimization Beyond the Datacenter*. 2015. arXiv: 1511.03575 [cs.LG].

[LSF21] Lin, Z., Sekar, V., and Fanti, G. "On the privacy properties of gan-generated samples". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1522–1530.

[Ma+23] Ma, C., Yuan, L., Han, L., Ding, M., Bhaskar, R., and Li, J. "Data Level Privacy Preserving: A Stochastic Perturbation Approach Based on Differential Privacy". In: *IEEE Transactions on Knowledge and Data Engineering* 35.4 (2023), pp. 3619–3631.

[MH20] Maaten, L. van der and Hannun, A. *The Trade-Offs of Private Prediction*. 2020. arXiv: 2007.05089 [cs.LG].

[McM+18] McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. *Learning Differentially Private Recurrent Language Models*. 2018. arXiv: 1710.06963 [cs.LG].

[MH19] Mo, F. and Haddadi, H. “Efficient and Private Federated Learning using TEE”. In: 2019. URL: https://api.semanticscholar.org/CorpusID:211627925.

[MM01] Moody, G. B. and Mark, R. G. “The impact of the MIT-BIH arrhythmia database”. In: *IEEE engineering in medicine and biology magazine* 20.3 (2001), pp. 45–50.

[NS08] Narayanan, A. and Shmatikov, V. “Robust De-anonymization of Large Sparse Datasets”. In: *2008 IEEE Symposium on Security and Privacy (sp 2008)*. 2008, pp. 111–125.

[Pap+17] Papernot, N., Abadi, M., Erlingsson, Ú., Goodfellow, I., and Talwar, K. *Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data*. 2017. arXiv: 1610.05755 [stat.ML].

[Pei+21] Pei, H., Ren, K., Yang, Y., Liu, C., Qin, T., and Li, D. “Towards generating real-world time series data”. In: *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2021, pp. 469–478.

[Pha+17] Phan, N., Wu, X., Hu, H., and Dou, D. “Adaptive Laplace Mechanism: Differential Privacy Preservation in Deep Learning”. In: *2017 IEEE International Conference on Data Mining (ICDM)*. 2017, pp. 385–394.

[RR07] Rahimi, A. and Recht, B. “Random Features for Large-Scale Kernel Machines”. In: *Advances in Neural Information Processing Systems*. Ed. by Platt, J., Koller, D., Singer, Y., and Roweis, S. Vol. 20. Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

[SWP22a] Schmidl, S., Wenig, P., and Papenbrock, T. “Anomaly detection in time series: a comprehensive evaluation”. In: *Proc. VLDB Endow.* 15.9 (May 2022), pp. 1779–1797.

[SWP22b] Schmidl, S., Wenig, P., and Papenbrock, T. “Anomaly detection in time series: a comprehensive evaluation”. In: *Proceedings of the VLDB Endowment* 15.9 (2022), pp. 1779–1797.

[SHS01] Scholkopf, B., Herbrich, R., and Smola, A. “A Generalized Representer Theorem”. In: *COLT/EuroCOLT*. 2001. URL: https://api.semanticscholar.org/CorpusID:9256459.

[SSJ18] Shailaja, K., Seetharamulu, B., and Jabbar, M. "Machine learning in healthcare: A review". In: *2018 Second international conference on electronics, communication and aerospace technology (ICECA)*. IEEE. 2018, pp. 910–914.

[Sho+17] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. *Membership Inference Attacks against Machine Learning Models*. 2017. arXiv: 1610.05820 [cs.CR].

[SOT22] Stadler, T., Oprisanu, B., and Troncoso, C. *Synthetic Data – Anonymisation Groundhog Day*. 2022. arXiv: 2011.07018 [cs.LG].

[13] "Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms". In: *ANSI/AAMI EC57:2012/(R)2020*. 2013. URL: https://array.aami.org/doi/abs/10.2345/9781570204784.ch1.

[WGW20] Wang, H., Ge, Z., and Wang, Z. "Accurate ECG data generation with a simple generative adversarial network". In: *Journal of Physics: Conference Series*. Vol. 1631. 1. IOP Publishing. 2020, p. 012073.

[Wan+18] Wang, L., Huang, K., Sun, K., Wang, W., Tian, C., Xie, L., and Gu, Q. "Unlock with Your Heart: Heartbeat-Based Authentication on Commercial Mobile Phones". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2.3 (Sept. 2018). URL: https://doi.org/10.1145/3264950.

[Wan+23] Wang, Y., Wang, Q., Zhao, L., and Wang, C. "Differential privacy in deep learning: Privacy and beyond". In: *Future Generation Computer Systems* (2023).

[War65] Warner, S. L. "Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias". In: *Journal of the American Statistical Association* 60.309 (1965). PMID: 12261830, pp. 63–69.

[WS18] Wiens, J. and Shenoy, E. S. "Machine learning for healthcare: on the verge of a major shift in healthcare epidemiology". In: *Clinical infectious diseases* 66.1 (2018), pp. 149–153.

[Xie+23] Xie, C., McCullum, L., Johnson, A., Pollard, T., Gow, B., and Moody, B. *Waveform Database Software Package for Python*. Version 4.1.0. Jan. 24, 2023. URL: https://physionet.org/content/wfdb-python/4.1.0/.

[Xie+18] Xie, L., Lin, K., Wang, S., Wang, F., and Zhou, J. *Differentially Private Generative Adversarial Network*. 2018. arXiv: 1802.06739 [cs.LG].

[Zha+12] Zhang, J., Zhang, Z., Xiao, X., Yang, Y., and Winslett, M. *Functional Mechanism: Regression Analysis under Differential Privacy*. 2012. arXiv: 1208.0219 [cs.DB].

[ZCZ19a] Zhao, J., Chen, Y., and Zhang, W. "Differential Privacy Preservation in Deep Learning: Challenges, Opportunities and Solutions". In: *IEEE Access* 7 (2019), pp. 48901–48911.

[ZCZ19b] Zhao, J., Chen, Y., and Zhang, W. "Differential privacy preservation in deep learning: Challenges, opportunities and solutions". In: *IEEE Access* 7 (2019), pp. 48901–48911.

[Zhu+19] Zhu, F., Ye, F., Fu, Y., Liu, Q., and Shen, B. "Electrocardiogram generation with a bidirectional LSTM-CNN generative adversarial network". In: *Scientific reports* 9.1 (2019), p. 6734.