TraVaG: Differentially Private Trace Variant Generation Using GANs

Majid Rafiei [iD] [✉], Frederik Wangelik [iD] , Mahsa Pourbafrani [iD], and Wil M.P. van der Aalst [iD]

Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany

Abstract. Process mining is rapidly growing in the industry. Consequently, privacy concerns regarding sensitive and private information included in event data, used by process mining algorithms, are becoming increasingly relevant. State-of-the-art research mainly focuses on providing privacy guarantees, e.g., differential privacy, for trace variants that are used by the main process mining techniques, e.g., process discovery. However, privacy preservation techniques for releasing trace variants still do not fulfill all the requirements of industry-scale usage. Moreover, providing privacy guarantees when there exists a high rate of infrequent trace variants is still a challenge. In this paper, we introduce TraVaG as a new approach for releasing differentially private trace variants based on Generative Adversarial Networks (GANs) that provides industry-scale benefits and enhances the level of privacy guarantees when there exists a high ratio of infrequent variants. Moreover, TraVaG overcomes shortcomings of conventional privacy preservation techniques such as bounding the length of variants and introducing fake variants. Experimental results on real-life event data show that our approach outperforms state-of-the-art techniques in terms of privacy guarantees, plain data utility preservation, and result utility preservation.

Keywords: Process Mining · Event Data · Differential Privacy · GANs · Machine Learning · Autoencoder

1 Introduction

Process mining is a family of data-driven techniques for business process discovery, analysis, and improvement. Process mining techniques require event data, which are widely available in most information systems, including ERP, SCM, and CRM systems. During the last decade, process mining has been successfully deployed in many industries, and it has become a crucial success factor for any type of business. Similar to any data-driven technique in the larger area of data science, concerns about the privacy of people whose data are processed by process mining algorithms are developing as the amount of event data and their usage rise. Thus, privacy regulations, e.g., GDPR [1], restrict data storage and process, which motivates the development of privacy preservation techniques.

Modern privacy preservation methods are mostly based on Differential Privacy (DP), which provides a privacy definition by introducing noise into data.

Table 1: A simple event log from the healthcare context, including trace variants and their frequencies.

Trace Variant	Frequency
$\langle register, visit, blood\text{-}test, visit, release \rangle$	15
$\langle register, blood\text{-}test, visit, release \rangle$	12
$\langle register, visit, hospitalization, surgery, release \rangle$	5
$\langle register, visit, blood\text{-}test, blood\text{-}test, release \rangle$	2

This is because of its significant properties, including its ability to ensure mathematically proven privacy and protect against PSO (predicate-singling-out) attacks [6]. The purpose of DP-based approaches is to inject noise into the released output in order to conceal the involvement of an individual. State-of-the-art research in process mining leveraging privacy preservation techniques based on DP focuses on releasing distributions of trace variants, which serve as the foundation for core process mining techniques such as process discovery and conformance checking [2]. A trace variant refers to a complete sequence of activities performed for an individual that is considered to be sensitive and private information. In the healthcare context, for instance, a trace variant shows a complete sequence of treatment-related activities performed for a patient that is private information itself and can also be exploited to conclude other sensitive information, e.g., the disease of the patient. Table 1 shows a small sample of a trace variant distribution in the healthcare context. Note that in a trace variant distribution, each trace variant is associated with an individual, a so-called case. Moreover, each case has precisely one trace variant.

To achieve DP for trace variants, conventional so-called *prefix-based* approaches inject noise drawn from a *Laplacian distribution* into the variant distribution obtained from an event log [11,21]. These approaches need to generate all possible unique variants based on a set of activities to provide differential privacy for the original distribution of variants. Since the set of possible variants that can be generated given a set of activities is infinite, prefix-based techniques need to limit the length of generated sequences. Also, to limit the search space, these approaches typically include a pruning parameter to exclude less frequent prefixes. Such a process to obtain DP has a high computational complexity and results in the following drawbacks: (1) *introducing fake variants*, (2) *removing frequent true variants*, and (3) *having limited length for generated variants*.

Several approaches have been proposed to partially or entirely address the aforementioned drawbacks. A method, called SaCoFa [11], aims to mitigate drawbacks (1) and (2) by gaining knowledge regarding the underlying process semantics from the original event data. However, the privacy quantification of all extra queries to gain knowledge regarding the underlying semantics is not discussed. Moreover, the third drawback still remains since this work itself is a prefix-based approach. In [10] and a technique called Libra [9], which is based on [10], trace variants are converted to a DAFSA (Deterministic Acyclic Finite State Automata) representation to avoid such drawbacks. However, Libra introduces a clipping parameter for removing infrequent variants. This clipping parameter grows based on the number of unique trace variants and the strength of privacy guarantees. Thus, depending on the number of unique trace variants and privacy

parameters, Libra may even remove all the variants and return empty outputs. A recent work called TraVaS [25] proposes an approach based on *differentially private partition selection strategies* to overcome the above-mentioned drawbacks. Similar to Libra, TraVaS also removes infrequent trace variants. However, in TraVaS, the threshold for removing infrequent variants is only dependent on the input privacy parameters and does not grow with the number of unique variants or the size of event data. Yet, for small event data with a high rate of unique trace variants, TraVaS may not be able to provide strong privacy guarantees.

In this paper, we introduce TraVaG to generate differentially private trace variants from an original variant distribution by means of GANs (Generative Adversarial Networks) [13]. The main idea of TraVaG is to privately learn important event data characteristics. The trained GAN enables the generation of new synthetic anonymized variants that are statistically similar to the original data. Trained generative models work without data access. Thus, as long as the statistical characteristics of the original data do not significantly change, one does not need to apply DP directly to the original event data. For industry-scale big event data, this property can considerably improve the computational complexities [22]. Moreover, TraVaG is based on DP-SGD (Differentially Private - Stochastic Gradient Descent) [3] optimization techniques that avoid thresholding on training data or released network outputs. Hence, TraVaG can generate infinite and arbitrarily large anonymized synthetic trace variants even if the original variant frequencies are comparably small. Moreover, our experiments on real-life event logs demonstrate a better performance of TraVaG compared to state-of-the-art techniques in terms of data utility preservation for the same privacy guarantees.

The remainder of this paper is structured as follows. In Section 2, we provide a summary of related work. Preliminaries are provided in Section 3. In Section 4, we present the details of TraVaG. Section 5 discusses the experimental results based on real-life event logs, and Section 6 concludes the paper.

2   Related Work

Privacy-preserving process mining is recently growing in importance. Several techniques have been proposed to address privacy issues in process mining. In the following, we provide a summary of the work focusing on *releasing differentially private event data* and *generating differentially private event data*.

2.1   Releasing Differentially Private Event Data

In [21], the authors apply an (ϵ, δ)-DP mechanism to event logs to privatize *directly-follows relations* and trace variants. The underlying principle uses a combination of an (ϵ, δ)-DP noise generator and an iterative query engine that allows an anonymized publication of trace variants with an upper bound on their length. In [11], SaCoFa has been introduced as an extension of [21], where the goal is to optimize the query structures with the help of underlying semantics. All the aforementioned techniques follow the so-called prefix-based approach that suffers

from the drawbacks explained in Section 1. To deal with such drawbacks, in [10], the authors introduced an approach that transforms a trace variant distribution into a DAFSA representation. This approach aims to keep all the original trace variants that may result in high noise injection during the anonymization process. Libra [9] is a recent work that employs the approach proposed in [10] and aims to increase utility using subsampling and composing privatized subsamples to release differentially private event data. TraVaS [25] introduces a novel approach based on differentially private partition selection to address the mentioned drawbacks in Section 1.

2.2   Generating Differentially Private Synthetic Data

Although DP-based generative Artificial Neural Networks (ANNs) have been quite extensively researched in the major field of data science and machine learning, they have not been used in the context of process mining. Thus, we mainly focus on some of the work outside the domain of process mining. In [5], the authors adopted a so-called *variational autoencoder*, DP-VAE, which assumes that the mapping from real data to the Gaussian distribution can be efficiently learned. A different direction was then chosen by [12], where the authors used a *Wasserstein* GAN (WGAN) to generate differentially private mixed-type synthetic outputs employing a Wasserstein-distance-based loss function. Finally, in [27], the concepts of WGAN and DP-VAE were combined to first learn a private data encoding and then generate respective encoded data. We adapted this principle for our work to cope with the large dimensionality of event data.

Research in non-private generative models for process mining, primarily focuses on exploiting ANNs and GANs to predict the next state of processes such as [17], and [19]. Note that the approach in [17] only provides synthetic event data without any privacy guarantees.

3   Preliminaries

We start the preliminaries by introducing basic notations and mathematical concepts. Let A be a set. $B(A)$ is the set of all multisets over A. Given B_1 and B_2 as two multisets, $B_1 \uplus B_2$ is the sum over multisets, e.g., $[a^2, b^3] \uplus [b^2, c^2] = [a^2, b^5, c^2]$. We define a finite sequence over A of length n as $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$ where $\sigma(i) = a_i \in A$ for all $i \in \{1, 2, \ldots, n\}$. The set of all finite sequences over A is denoted with A^*.

3.1   Event Log

Process mining techniques employ event data that are typically collections of unique events recorded per activity execution and characterized by their attributes, e.g., *activity* and *timestamp*. Events in an event log have to be unique. A *trace* is a single process execution represented as a sequence of events belonging to a case (individual) and having a fixed ordering based on timestamps. An

event cannot appear in more than one trace or multiple times in one trace. Our work focuses on the control-flow aspect of an event log that only considers the activity attribute of events in a trace, so-called a *trace variant*. Thus, we define a simple event log based on activity sequences, so-called *trace variants*.

Definition 1 (Simple Event Log). *Let $\mathcal{A}$ be the universe of activities. A simple event log L is defined as a multiset of trace variants $\mathcal{A}^*$, i.e., $L \in B(\mathcal{A}^*)$. $\mathcal{L}$ denotes the universe of simple event logs.*

In a simple event log representing a distribution of trace variants, one case, which refers to an individual, cannot contribute to more than one trace variant.

3.2   Differential Privacy (DP)

The main idea of DP is to inject noise into the original data in such a way that an observer who sees the randomized output cannot with certainty tell if the information of a specific individual is included in the data [8]. Considering simple event logs, as our sensitive event data, we define differential privacy in Definition 2.

Definition 2 ((ϵ,δ)-DP for Event Logs). *Let L_1 and L_2 be two neighboring event logs that differ only in a single entry, i.e., $L_2 = L_1 \uplus [\sigma]$ for any $\sigma \in \mathcal{A}^*$. Also, let $\epsilon \in \mathbb{R}_{>0}$ and $\delta \in \mathbb{R}_{>0}$ be two privacy parameters. A randomized mechanism $\mathcal{M}_{\epsilon,\delta}: \mathcal{L} \to \mathcal{L}$ provides (ϵ, δ)-DP if for all $S \subseteq B(\mathcal{A}^*)$: $Pr[\mathcal{M}_{\epsilon,\delta}(L_1) \in S] \le e^{\epsilon} \times Pr[\mathcal{M}_{\epsilon,\delta}(L_2) \in S] + \delta$.*

In Definition 2, ϵ specifies the probability ratio, and δ allows for a linear violation. In the strict case of $\delta = 0$, $\mathcal{M}$ offers ϵ-DP. The randomness of respective mechanisms is typically ensured by the noise drawn from a probability distribution that perturbs original variant-frequency tuples and results in non-deterministic outputs. The smaller the privacy parameters are set, the more noise is injected into the mechanism outputs, entailing a decreasing likelihood of tracing back the instance existence based on outputs.

3.3   Generative Adversarial Networks (GANs)

A generative adversarial network (GAN) represents a special type of ANN compound to synthesize similar data to its original input. It comprises two separate ANNs, a *generator* and a *discriminator* [13]. The training principle follows a two-player game: a generator tries to fool the discriminator by generating authentic fake data while a discriminator tries to distinguish between real and fake results. A generator $gen : \mathbb{Z}^m \to \mathbb{R}^n$ and a discriminator $dis : \mathbb{R}^n \to \{0, 1\}$ can be described as highly parametrizable functions. Here, a generator gen is seeded with random multivariate Gaussian noise $z \in Z^m$ of user-defined dimension m that is translated into a synthetic desired output. A discriminator dis aims to determine whether its input originates from the generator's output. In a simple form, it outputs a binary decision variable, where 0 means the input is fake and 1 means the input is original data. In our work, we apply a GAN architecture to synthesize event data.

3.4   Autoencoder

An *autoencoder* is a certain type of ANN structure used to learn efficient encodings of unlabeled data [15]. The respective encoding is validated and optimized by attempting to regenerate the input from the encoding by decoding. The autoencoder learns the encoding for a set of data to typically provide dimensionality reduction. As a result, an autoencoder always consists of two separate ANNs, an encoder $enc : \mathbb{R}^n \rightarrow \mathbb{R}^d$ and a decoder $dec : \mathbb{R}^d \rightarrow \mathbb{R}^n$. These components allow for transforming high-dimensional data $x \in \mathbb{R}^n$ to a compact representation within the so-called latent space $\mathbb{R}^d$ and vice versa (typically $d \ll n$). The specific mappings of enc and dec are characterized by the network's weights and learned from the data during the training phase. For our work, we employ an autoencoder structure to achieve a compressed encoding of input event data.

4   TraVaG

As presented in Section 2, DP-based generative networks have been extensively researched outside of the process mining context. Typical approaches either adopt variational autoencoder architectures that leverage both encoder and decoder components or GAN architectures employing a discriminator and a generator part. When transferring these ideas to event data, one crucial aspect is the high-dimensional structure that turns out to be challenging during training, particularly if strong DP is added. Thus, we follow the approach of the novel work [27] and [14] that combines the compression functionality of autoencoders with the flexibility of GANs and demonstrated superior performance for general high-dimensional mix-type input data [27]. Instead of directly generating new event logs, we first learn a compressed encoding and then train a GAN to reproduce data within the encoded latent space. Final datasets are obtained by decoding back the dimension-reduced intermediate format. This principle mitigates the complication of GANs when extracting statistical properties from feature-rich data that is limited in size. Particularly, sparse features can be compressed without significant loss of information, while generator networks improve their learning performance due to the lower dimension. Moreover, no Gaussian Mixture distribution is enforced on the latent space, as is the case for typical generative stand-alone autoencoder methods [5].

4.1   The TraVaG Framework

Different components and the workflow of our framework are shown in Figure 1. We start with preprocessing a simple event log that contains variant distributions in the form of variant-frequency pairs. There are two common possibilities. The first option considers the activities within variants and extracts all subsequences of direct neighbors, i.e., Directly-Follows Relations (DFRs). These DFRs are then mapped to a binary or number space and either fed into a GAN as a single feature or as two features along with their frequencies. A downside of this
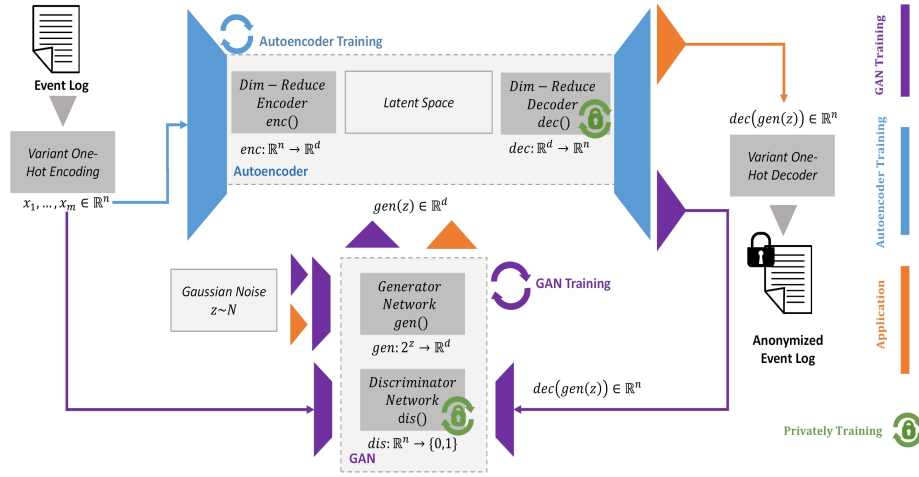
Fig. 1: A simplified workflow diagram of the TraVaG training and application processes.

method is that the generator serves as a sequence constructor which allows the creation of artificial variants in the postprocessing phase where all generated activity pairs are linked back together. To avoid creating fake trace variants, we choose the second option, where only complete variants are considered as inputs. Therefore, a simple event log L with n variants and m cases is binary-encoded as follows. Within a $m \times n$ matrix, each variant represents a binary feature column and each case denotes a row instance that contains 1 at the respective variant column and 0 elsewhere (sparse matrix). Analogously, this transformation can be inverted back to the original data space. Thus, TraVaG never produces fake trace variants. Also, one-hot encoding does not influence the data statistics and hence does not incur any privacy costs. We refer to this preprocessing procedure as *one-hot encoding* and *one-hot decoding*.

We perform two main training phases including autoencoder training (blue parts) and GANs training (purple parts). Since the focus of this work is on the privacy aspect, we describe the privately trained components in more detail. A detailed algorithmic explanation of the training components including the structure of the networks, parameter tuning, activation functions, loss measures, and optimizations is provided in our supplementary document.[1] After the preprocessing, the sparse binary variant vectors $x_1 \ldots x_m {\in} \mathbb{R}^n$ are forwarded to the autoencoder training phase, including an encoder and a decoder component. These components allow for transforming high-dimensional data $x_i {\in} \mathbb{R}^n$ to a compact representation within the so-called latent space $\mathbb{R}^d$ and vice versa, s.t., $d \ll n$. The dimension d is a hyperparameter of the autoencoder and needs to be selected w.r.t. the GANs configuration. Since the encoder does not participate in the process of training the GAN or synthesizing new event data, it does not need to be optimized privately [4, 5]. The decoder is strongly involved in the

---

[1] `https://github.com/wangelik/TraVaG/blob/main/supplementary/TraVaG.pdf`

anonymization process and is released to the public. Thus, the training of the decoder is performed privately by means of DP-SGD (see Section 4.2).

The same one-hot encoded data $x_1 \dots x_m \in \mathbb{R}^n$ are used to train a GAN consisting of two feed-forward ANNs; a generator $gen : 2^{\mathbb{Z}} \to \mathbb{R}^d$ and a discriminator $dis : \mathbb{R}^n \to \{0, 1\}$. The goal of the generator gen is to construct synthetic data within the output space $\mathbb{R}^d$ that are similar to the compressed variants. It is seeded with random multivariate Gaussian noise z of a user-defined dimension. The discriminator dis aims at determining whether its input originates from the decompressed generator output $dec(gen(.))$ or from the original data source $x_i, 1 \leq i \leq m$. Both components are parameterized by their network weights and trained iteratively to outplay each other. Whereas the generator attempts to find latent space outputs that are hard to distinguish from real encoded data by the discriminator, the latter tries to expose these synthetic data records. Eventually, this principle enables the generator to learn and capture the statistical properties of the input variant distribution through the lenses of the autoencoder. Note that due to the integrated autoencoder, the generator only targets the latent space $\mathbb{R}^d$ which is much easier to achieve than constructing data in $\mathbb{R}^n$. Also, it averts to access the real confidential data space and does not need to be trained with DP as opposed to the discriminator that is again privately optimized with DP-SGD algorithms [27].

Once both the autoencoder and GAN are trained, one can generate new synthetic anonymized event data (orange parts). The underlying mechanism equals the training step of the generator. Starting with a random Gaussian noise sample z, this noise becomes digested by the generator, yielding $gen(z)$. From the latent space, the decoder then maps $gen(z)$ to $dec(gen(z))$. Finally, the synthetic one-hot encoded result is transformed back to the variant universe. One compelling advantage of TraVaG lies in the underlying data format. Since the feature space represents the different variants of the original data, TraVaG considers them as given and only has to learn their distribution during training. When applied, the framework reconstructs an anonymized version of this distribution over multiple runs without introducing new variants. The more synthetic data are created, the better the consolidated TraVaG output, i.e., new anonymized variants better approximate the original variant distribution. Note that this process does not converge to the true variant frequencies, but to the TraVaG-internal learned anonymous version. Thus, it is recommended to run TraVaG at least as often as the number of cases in the original event log. In case smaller privatized datasets are needed, the output can be down-sampled during postprocessing rounds.

4.2   Differentially Private - Stochastic Gradient Descent (DP-SGD)

To render SGD differentially private, Abadi et al. [3] proposed the following two steps. Given a dataset $X = \{x_i \in \mathbb{R}^n \mid 1 \leq i \leq m\}$, f as a loss function, and θ as the model parameter. First, the gradient $g_i = \nabla_\theta f_\theta(x_i)$ of each data sample x_i is clipped at some real value $C \in \mathbb{R}_{>0}$ to ensure its L^2-norm of the gradient does

not exceed the clipping value. For our work, we refer to the following clipping function[2]: $\text{clip}(g_i, C) = g_i \cdot \min(1, C/||g_i||_2)$.

Then, as Equation 1 shows, multivariate Gaussian noise parametrized by a noise multiplier $\Phi \in \mathbb{R}$ is added to the clipped gradient vectors before averaging over the batch $B \subseteq X$. We further denote the identity matrix as I and the Gaussian distribution of unspecified dimension as $\mathcal{N}$.

$$g_B \leftarrow \frac{1}{|B|} \left(\sum_{i \in B} \text{clip}(\nabla_\theta f_\theta(x_i), C) + \mathcal{N}(0, C^2 \Phi^2 I) \right) \tag{1}$$

The noisy-clipped-averaged gradient g_B is now differentially private and can be used for conventional descent steps: $\theta \leftarrow \theta - \eta \cdot g_B$, where η is the so-called *learning rate*. Note that clipping the individual gradients as in Equation 1 can also be replaced by instead clipping gradients of groups of more data points, so-called *microbatches* [22]. Instead of the common DP parameters ϵ and δ, DP-SGD uses the related noise multiplier Φ. When translating between these two types of settings, novel research has demonstrated a tighter privacy bound if the batch sampling process for B is conducted according to a specific procedure [3]. This procedure independently selects each data point of X with a fixed probability q, the so-called *sampling rate*, in each step.

4.3   Privacy Accounting

To evaluate the exact privacy guarantee provided by DP-SGD algorithms, we employ the so-called *Renyi Differential Privacy* (RDP) [23], a different notion of DP typically used for private optimization. RDP is defined based on the concept of *Renyi divergence*. Given two probability distributions P and Q, the Renyi divergence of order α is defined as follows: $D_\alpha(P||Q) := \frac{1}{\alpha-1} \log \mathbb{E}_{x \sim Q} \left(\frac{P(x)}{Q(x)} \right)^\alpha$.

Definition 3 ((α, ϵ)-RDP for Event Logs). *Let L_1 and L_2 be two neighboring event logs that differ only in a single entry, e.g., $L_2 = L_1 \uplus [\sigma]$ for any $\sigma \in \mathcal{A}^*$. Given $\alpha > 1$ and $\epsilon \in \mathbb{R}_{>0}$, a randomized mechanism $\mathcal{M}_{\alpha,\epsilon}: \mathcal{L} \rightarrow \mathcal{L}$ provides (α, ϵ)-RDP if $D_\alpha(\mathcal{M}(L_1)||\mathcal{M}(L_2)) \leq \epsilon$.*

To obtain the final (ϵ, δ)-DP parameters, we employ the following two propositions on the composition of (α, ϵ)-RDP mechanisms and the conversion of (α, ϵ)-RDP parameters to (ϵ, δ)-DP parameters.

Proposition 1 (Composition of RDP [23]). *If $\mathcal{M}_1$ and $\mathcal{M}_2$ are two (α, ϵ_1)-RDP and (α, ϵ_2)-RDP mechanisms for $\alpha > 1$, respectively. Then, the composition of $\mathcal{M}_1$ and $\mathcal{M}_2$ satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

Proposition 2 (RDP Parameter Conversion [23]). *If a mechanism $\mathcal{M}$ satisfies (α, ϵ)-RDP with $\alpha > 1$, then for all $\delta > 0$, $\mathcal{M}$ satisfies $(\epsilon + (\log 1/\delta)/(\alpha-1), \delta)$-DP.*

During an iterative application of Gaussian mechanisms, as is the case in DP-SGD, the Renyi divergence allows more tightly capturing of the corresponding privacy loss than standard (ϵ, δ)-DP. To compute the final (ϵ, δ)-DP parameters from multiple runs of DP-SGD, the following three steps are followed.

---

[2] Note that also other clipping strategies exist, as highlighted in [22].

1. **Subsampled RDP.** Given a sampling rate q and noise multiplier Φ, the RDP privacy parameters for one iteration of DP-SGD can be derived as a non-explicit integral function of $\alpha \geq 1$ [23]. This function is standardized in many privacy-related optimization packages and will be referred to as $\text{RDP}_1(q, \Phi)$ [3].
2. **RDP Composition.** Since DP-SGD is most likely to run iteratively, we need to compose Step 1 over all executions according to Proposition 1. Hence, the resulting RDP parameters of T iterations are obtained by computing $\text{RDP}_T(q, \Phi, T) := \text{RDP}_1(q, \Phi) \cdot T$.
3. **Conversion to (ϵ, δ)-DP.** After retrieving an expression for the overall RDP privacy parameters with RDP_T, we need to convert the respective (α, ϵ) tuple to a (ϵ, δ) guarantee according to Proposition 2. Since the ϵ parameter of RDP is also a function of α, Step 3 involves optimizing for α to achieve a minimal ϵ and δ.

We apply this procedure to obtain the respective privacy guarantees (ϵ, δ)-DP on both the autoencoder and the GAN-based discriminator of TraVaG. The resulting values are then combined into a final privacy cost by the *composition theorem* of DP [8]. According to the composition theorem, different (ϵ, δ)-DP mechanisms can be easily combined into more complex algorithms at the cost of a directly measurable cumulative privacy loss, and the result still promises (ϵ, δ)-DP independent of the exact form of composition or query structure.

5   Experiments

We evaluate the performance of TraVaG on real-life event logs. We select two event logs of varying sizes and trace uniqueness. As we discussed in Section 1 and stated in other research such as [21], [11], and [9] infrequent variants are challenging to privatize. Thus, trace uniqueness is an important analysis criterion. The Sepsis log describes hospital processes for Sepsis patients and contains many rare traces [20]. In contrast, BPIC13 has significantly more cases at a four times smaller trace uniqueness [7]. BPIC13 describes an incident and problem management system called VINST. Both logs are realistic examples of confidential human-centered information where the case identifiers refer to individuals. Table 2 shows detailed log statistics.

We perform our evaluation for a wide range of the main privacy parameters $\epsilon \in \{0.01, 0.1, 1, 2\}$ and $\delta \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 0.01\}$. These ranges are selected in accordance with typical values employed at industrial applications as well as state-of-the-art DP research [9, 11, 21, 26]. We particularly note that extreme settings such as $\epsilon = 2, \delta = 0.5$ are not chosen due to practical relevance, but to demonstrate how the anonymization methods behave when starting from a weak- or non-private environment. Due to the probabilistic nature of (ϵ, δ)-DP, we run the TraVaG generator 100 times on all input event logs and all

Table 2: General statistics of the event logs used in our experiments.

Event Log	#Events	#Cases	#Activities	#Variants	Trace Uniqueness
Sepsis	15214	1050	16	846	80%
BPIC13	65533	7554	4	1511	20%

privacy parameters and report the average values. We compare our results with TraVaS [25] as a state-of-the-art technique and the original prefix-based framework called benchmark [21].[3] The sequence cutoff for the benchmark method is set to the length that covers 80% of variants in each log, and the remaining pruning parameter is adjusted such that on average anonymized logs contain a comparable number of variants with the given original log. The ANNs of TraVaG are configured by a semi-automated tuning approach w.r.t the different input logs. Whereas most design decisions and hyperparameters are tweaked according to results of manual tests as well as research experience, the settings: *batch size* (B), *number of iterations* (I) and *noise multiplier* (Φ) are automatically optimized via a grid-search [18] for fixed privacy levels. A detailed list of the derived settings for each event log, the concrete network designs, and configuration values are available on GitHub.[4]

5.1   Evaluation Measures

Suitable evaluation measures are required to assess the performance of an (ϵ, δ)-DP mechanism in terms of data (result) utility preservation. The *data utility* perspective measures the similarity between two logs independent of future applications. For evaluating data utility we employ the following measures: *relative log similarity* [24,25] and *absolute log difference* [25]. *Relative log similarity* measures the *earth mover's distance* between two trace variant distributions, where the normalized *Levenshtein* string edit distance is used as a similarity function between trace variants. This measure quantifies the degree to which the variant distribution of an anonymized log matches the original variant distribution on a scale from 0 to 1. *Absolute log difference* accounts for the situations where distribution-based measures provide misleading expressiveness [25]. Exemplary cases are event logs possessing similar variant distributions, but significantly different sizes. To calculate an absolute log difference value, we use the approach introduced in [25], where input logs are converted to a *bipartite graph* of variants as vertices. Then, a *cost network flow* problem is solved by setting demands and supplies to the absolute variant frequencies and utilizing a *Levenshtein* distance between variants as an edge cost. Thus, the result of this measure shows the minimal number of *Levenshtein* operations to transform variants of an anonymized log into variants of the original log. Details of the exact algorithms are available.[5]

We additionally evaluate the performance of TraVaG in terms of *result utility preservation* for *process discovery* as a specific application of trace variant distribution. In this respect, we use the *inductive miner infrequent* [16] with a default noise threshold of 20% to discover process models from the privatized event logs

---

[3] Note that in [25], TraVaS was already compared with SaCoFa [11] and benchmark [21] and showed better performance. Here, the benchmark method is included for easier comparison. Moreover, Libra [9] does not take ϵ as an input parameter but computes it based on α as an RDP parameter and its sampling strategy. This makes the comparison based on exact ϵ and δ parameters very difficult. Nevertheless, an important observation in contrast to TraVaG is that Libra returns an empty log for event logs with many infrequent variants, such as Sepsis when $\delta \leq 10^{-3}$.

[4] `https://github.com/wangelik/TraVaG/blob/main/supplementary/TraVaG.pdf`

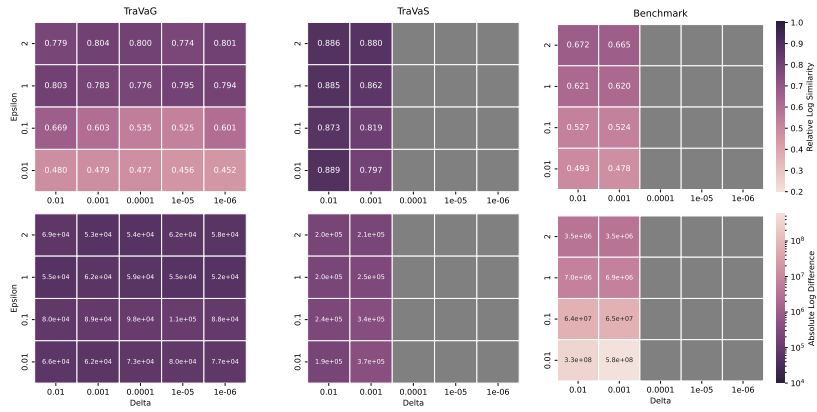[5] `https://github.com/wangelik/TraVaG/blob/main/supplementary/metrics.pdf`

Fig. 2: The *relative log similarity* and *absolute log difference* results of anonymized BPIC13 logs generated by TraVaG, TraVaS, and the benchmark method. Each value represents the mean of 100 generations for TraVaG and 10 algorithm runs for TraVaS and the benchmark method.

for all (ϵ, δ) settings under investigation. Then, we compare the models with the original event log to obtain token-based replay *fitness* and *precision* scores [2].

5.2   Data Utility Analysis

In this subsection, the results of the two aforementioned data utility metrics are presented for both real-life event logs. Figure 2 shows the average results on BPIC13 in a six-fold heatmap. The gray fields at the TraVaS and benchmark methods denote an unsuccessful algorithm execution. For $\delta < 10^{-3}$, the thresholding of TraVaS becomes too strict and removes many variants in the anonymized outputs. On the contrary, the benchmark method introduces artificial variants and noise to an extent that is unfeasible to average within reasonable time and accuracy. In opposition, TraVaG successfully manages to generate anonymized outputs for $\delta < 10^{-3}$. More importantly, both results of *relative log similarity* and *absolute log difference* do not illustrate clear decreasing trends on lower δ within the investigated parameter range. We explain this expected observation by the fact that TraVaG avoids any pruning mechanism on its output and implements less δ-dependent Gaussian noise via RDP into the gradients (see Section 4.3 and [23]).

Whereas the absolute log difference results maintain a rather stable output for the different (ϵ, δ) values, the TraVaG relative log similarity presents a strong positive ϵ-dependency. As a result, the absolute statistics (absolute Levenshtein distances and absolute frequencies) of the anonymized event data seem to be more similar to the original logs as the variant distributions. A rationale for this discrepancy lies in the still comparably small dataset with 7554 instances over 1511 variants (features). By construction, TraVaG accomplishes reproducing equally sized event logs containing many original variants but fails to pick up some characteristics of the underlying distribution once the input data or the
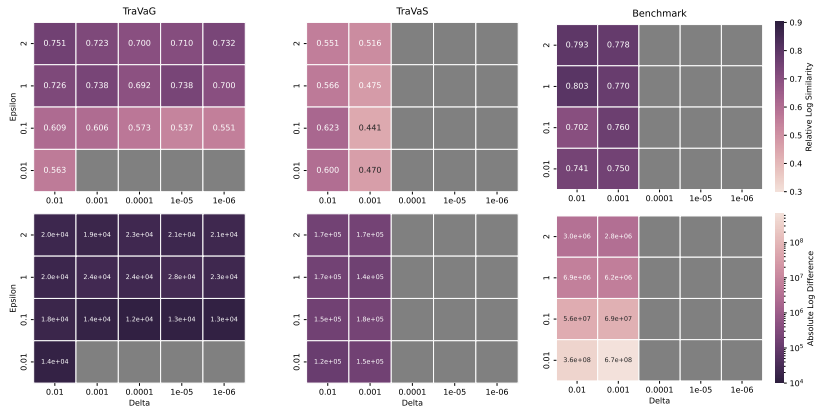
Fig. 3: The *relative log similarity* and *absolute log difference* results of anonymized Sepsis logs generated by TraVaG, TraVaS, and the benchmark method. Each value represents the mean of 100 generations for TraVaG and 10 algorithm runs for TraVaS and the benchmark method.

training iterations are limited. Hence, we expect this diverging trend to diminish with increasing training data.

The data utility results for the Sepsis log are presented in Figure 3. With only 1050 instances at 846 variants (features), this dataset is even smaller and thus more difficult to train for TraVaG than BPIC13. As a result, we observe similar, but more pronounced behavior of relative log similarity and absolute log difference metrics compared to Figure 2. An extreme example are the results at $\epsilon = 0.01, \delta < 10^{-2}$, where the introduced gradient noise turned out as too intense for the generative model to converge under the given training data size. For the remaining privacy settings, TraVaG again outperforms its competitors at the absolute log statistics while the relative log similarity performs slightly better than TraVaS and at the same order as the benchmark results for $\epsilon > 0.1$.

5.3 Process Discovery Analysis

Figure 4 illustrates the result utility analysis of TraVaG, TraVaS, and the benchmark on BPIC13. As discussed in Subsection 5.2, TraVaG successfully manages to produce results for $\delta < 10^{-3}$ where the other methods are not applicable. Except for the three outliers at $\epsilon = 0.1$, both fitness and precision show a stable distribution without considerable dependence on the different privacy parameters. In accordance with Figure 2, we thus conclude that the absolute log difference provides a better proxy for process-discovery-based performance of TraVaG than relative log similarity. Similarly, the strong scores on both metrics demonstrate a sufficient replay behavior between the model obtained from an anonymized log and the original log. Whereas fitness denotes that the process model still captures most of the real underlying event data, precision depicts only a small fraction of model decisions, not being included in the anonymized event log. Consequently, TraVaG accomplishes learning the most important facets of the BPIC13 variant distribution for the discovery algorithm to produce a fitted model. When com-
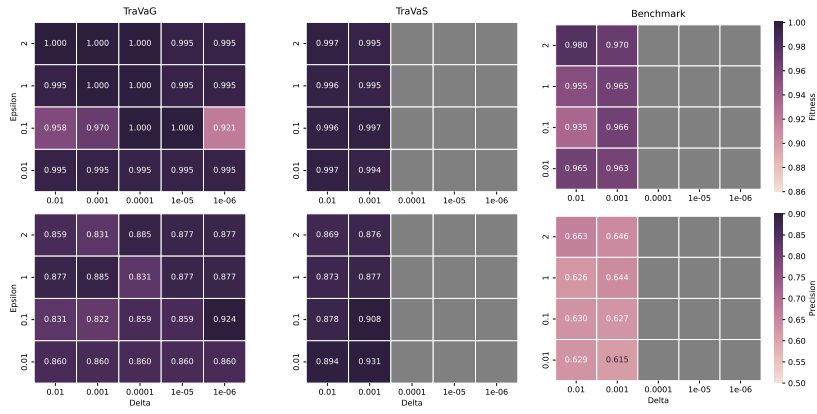
Fig. 4: The *fitness* and *precision* results of anonymized BPIC13 event logs generated by TraVaG, TraVaS, and the benchmark method. Each value represents the mean of 100 generations for TraVaG and 10 algorithm runs for TraVaS and the benchmark method.
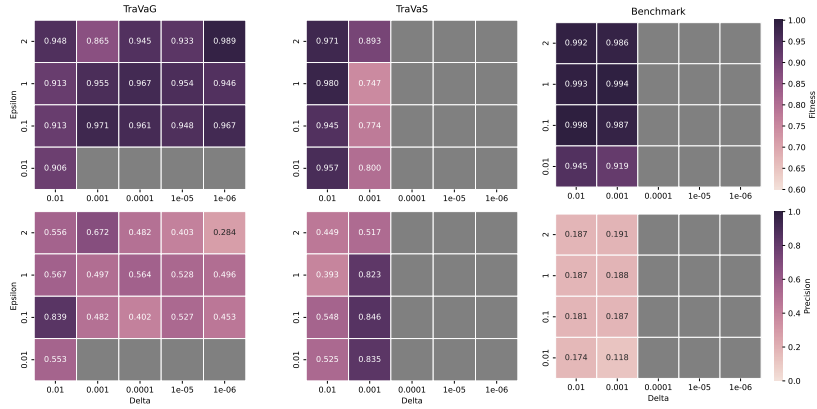

Fig. 5: The *fitness* and *precision* results of anonymized Sepsis event logs generated using TraVaG, TraVaS, and the benchmark method. Each value represents the mean of 100 generations for TraVaG and 10 algorithm runs for TraVaS and the benchmark method.

pared to the alternative methods, TraVaG achieves comparable scores as TraVaS and again outperforms the benchmark.

The result utility evaluation of the high trace-unique Sepsis log is presented in Figure 5. With respect to fitness, TraVaG shows similar values as TraVaS but a slight under-performance compared to the benchmark method. The main cause for this observation again refers to the infrequent variants and the small log size. While TraVaS maintains a strong δ-related threshold and TraVaG copes with the limited training data, the benchmark method introduces many artificial variants but tends to match the frequent traces. As a result, the discovered process models are able to replay most of the original behavior in contrast to TraVaG and TraVaS results. According to the aforementioned explanation, precision reflects an inverted trend. Here, the larger models of the benchmark method contain

many possible decision paths that are nonexistent in the underlying event log. For TraVaS and TraVaG, we thus achieve more precise anonymized process models.

6   Conclusion

TraVaG has shown that training a differentially private combination of autoencoders and GANs to synthesize anonymized event data from an underlying original variant distribution outperforms current state-of-the-art selection-based variant anonymization techniques and prefix-based approaches. Particularly, for strong privacy at the low δ range. Moreover, TraVaG has the unique advantages of outstanding resource-efficient execution, the absence of distorting noise thresholds, a general acceptance of continuous data streams, and no fake variant generation. In combination, these characteristics allow TraVaG to efficiently operate with infrequent variant data in the low δ regime without real competitors. Nevertheless, we note that the framework comprises a more complex training procedure and privacy budget accounting than approaches that directly digest DP parameters, such as TraVaS [25]. We have to follow the one-way procedure to first obtain RDP parameters (ϵ, α) from noise multiplier Φ, sampling rate q, iterations T and then convert (ϵ, α) to (ϵ, δ). Note that a similar procedure is followed by other techniques that are based on RDP, such as Libra [9]. Consequently, specific privacy levels can only be ensured by repeatedly analyzing different TraVaG network settings until a successful match is found. This hyperparameter dependence could be studied in more detail and even coupled with a fully automated tuning strategy in future work.

References

1. GDPR, `http://data.europa.eu/eli/reg/2016/679/oj`, Accessed: 2022-12-15
2. van der Aalst, W.M.P.: Process Mining - Data Science in Action, Second Edition. Springer (2016)
3. Abadi, M., Chu, A., Goodfellow, I.J., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 308–318. ACM (2016)
4. Ács, G., Melis, L., Castelluccia, C., Cristofaro, E.D.: Differentially private mixture of generative neural networks. IEEE Trans. Knowl. Data Eng. **31**(6) (2019)
5. Chen, Q., Xiang, C., Xue, M., Li, B., Borisov, N., Kaafar, D., Zhu, H.: Differentially private data generative models. CoRR **abs/1812.02274** (2018)
6. Cohen, A., Nissim, K.: Towards formalizing the gdpr's notion of singling out. Proc. Natl. Acad. Sci. USA **117**(15), 8344–8352 (2020)
7. van Dongen, B.F., Weber, B., Ferreira, D.R., Weerdt, J.D.: BPI challenge 2013. In: Proceedings of the 3rd Business Process Intelligence Challenge (2013)
8. Dwork, C.: Differential privacy: A survey of results. In: Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings. vol. 4978, pp. 1–19. Springer (2008)
9. Elkoumy, G., Dumas, M.: Libra: High-utility anonymization of event logs for process mining via subsampling. In: 4th International Conference on Process Mining, ICPM. IEEE (2022)

10. Elkoumy, G., Pankova, A., Dumas, M.: Mine me but don't single me out: Differentially private event logs for process mining. In: 3rd International Conference on Process Mining, ICPM 2021,. pp. 80–87. IEEE (2021)
11. Fahrenkrog-Petersen, S.A., Kabierski, M., Rösel, F., van der Aa, H., Weidlich, M.: Sacofa: Semantics-aware control-flow anonymization for process mining. In: 3rd International Conference on Process Mining, ICPM 2021, Eindhoven, The Netherlands, October 31 - Nov. 4, 2021. pp. 72–79. IEEE (2021)
12. Frigerio, L., de Oliveira, A.S., Gomez, L., Duverger, P.: Differentially private generative adversarial networks for time dummy-series, continuous, and discrete open data. In: ICT Systems Security and Privacy Protection - 34th IFIP TC 11 International Conference, SEC 2019. vol. 562. Springer (2019)
13. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2020)
14. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (2017)
15. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, Conference Track Proceedings (2014)
16. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from incomplete event logs. In: Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings. vol. 8489, pp. 91–110. Springer (2014)
17. Li, K., Yang, S., Sullivan, T.M., Burd, R.S., Marsic, I.: Generating privacy-preserving process data with deep generative models. CoRR **abs/2203.07949** (2022)
18. Liashchynskyi, P., Liashchynskyi, P.: Grid search, random search, genetic algorithm: A big comparison for NAS. CoRR **abs/1912.06059** (2019)
19. Lu, Y., Chen, Q., Poon, S.K.: A deep learning approach for repairing missing activity labels in event logs for process mining. Inf. **13**(5), 234 (2022)
20. Mannhardt, F.: Sepsis Cases (2016). https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460
21. Mannhardt, F., Koschmider, A., Baracaldo, N., Weidlich, M., Michael, J.: Privacy-preserving process mining - differential privacy for event logs. Bus. Inf. Syst. Eng. **61**(5), 595–614 (2019)
22. McMahan, H.B., Andrew, G.: A general approach to adding differential privacy to iterative training procedures. CoRR **abs/1812.06210** (2018)
23. Mironov, I.: Rényi differential privacy. In: 30th IEEE Computer Security Foundations Symposium, CSF 2017. pp. 263–275. IEEE Computer Society (2017)
24. Rafiei, M., van der Aalst, W.M.P.: Towards quantifying privacy in process mining. In: Process Mining Workshops - ICPM 2020 International Workshops. vol. 406, pp. 385–397. Springer (2020)
25. Rafiei, M., Wangelik, F., van der Aalst, W.M.P.: TraVaS: differentially private trace variant selection for process mining. In: Process Mining Workshops - ICPM 2022 International Workshops. Springer (2022)
26. Tang, J., Korolova, A., Bai, X., Wang, X., Wang, X.: Privacy loss in apple's implementation of differential privacy on macos 10.12. CoRR **abs/1709.02753** (2017)
27. Tantipongpipat, U.T., Waites, C., Boob, D., Siva, A.A., Cummings, R.: Differentially private synthetic mixed-type data generation for unsupervised learning. Intell. Decis. Technol. **15**(4), 779–807 (2021)