PART-GAN: Privacy-Preserving Time-Series Sharing

Shuo Wang[1,2(✉)], Carsten Rudolph[1], Surya Nepal[2], Marthie Grobler[2], and Shangyu Chen[3]

[1] Monash University, Melbourne, Australia
{shuo.wang,carsten.rudolph}@monash.edu
[2] CSIRO, Melbourne, Australia
{surya.nepal,marthie.grobler}@csiro.au
[3] University of Melbourne, Melbourne, Australia
shangyuc@student.unimelb.edu.au

Abstract. In this paper, we provide a practical privacy-preserving generative model for time series data augmentation and sharing, called PART-GAN. Our model enables the local data curator to provide a freely accessible public generative model derived from original data for cloud storage. Compared with existing approaches, PART-GAN has three key advantages: It enables the generation of an unlimited amount of synthetic time series data under the guidance of a given classification label and addresses the incomplete and temporal irregularity issues. It provides a robust privacy guarantee that satisfies differential privacy to time series data augmentation and sharing. It addresses the trade-offs between utility and privacy by applying optimization strategies. We evaluate and report the utility and efficacy of PART-GAN through extensive empirical evaluations of real-world health/medical datasets. Even at a higher level of privacy protection, our method outperforms GAN with ordinary perturbation. It achieves similar performance with GAN without perturbation in terms of inception score, machine learning score similarity, and distance-based evaluations.

Keywords: Generative model · Data sharing · Privacy-preserving · Deep learning · Differential privacy

1 Introduction

Multivariate time series data are ubiquitous in various fields of study from health and medical records to financial and meteorological observations. Time series data generally contain a large number of missing values due to the intermittent failures/faults of collection devices. Besides, the time intervals of the observations in time series are not always fixed. Current studies have started to synthesize time series data using both synthesizers and machine learning models to some degree of success, e.g., Generative Adversarial Network (GAN) [12–14].

I. Farkaš et al. (Eds.): ICANN 2020, LNCS 12396, pp. 578–593, 2020.
https://doi.org/10.1007/978-3-030-61609-0_46

GANs had brought the practical potential to relieve the data availability limitation and showed significant progress in generating authentic-looking data. However, it has been demonstrated that releasing only the generative distribution derived from generative models can reveal private information of the original training samples [2]. It is necessary to embed privacy-preserving mechanisms into the generative models to break privacy barriers that hamper data sharing.

In this paper, we propose a Privacy-preserving Augmentation and Releasing scheme for Time series data via GAN (PART-GAN), which adopts a robust differential privacy paradigm and incorporates three optimizing strategies to improve utility and feasibility. Our contributions can be summarized as follows:

(1) We propose the Conditional & Temporal - Generative Adversarial Network (CT-GAN) to generate new noisy and irregularly-sampled time series data. This augments the unbalanced time-series datasets for better classification performance. The generator and the discriminator are conditioned with the timestamps to capture the latent representations of the time series for naturally mimicking the time-series dynamics and according to a given label (e.g., arrhythmia classification).
(2) Based on CT-GAN, we propose a PART-GAN scheme that can be applied to produce an unlimited amount of synthetic privacy-sensitive time-series data for many participants to balance their local datasets for the deep neural network training. This is done in a robust privacy-preserving manner, integrated with differential privacy mechanisms and resulting in a better classification performance after data augmentation.
(3) To the best of our knowledge, PART-GAN is the first attempt to balance the utility and privacy trade-offs the privacy-preserving model releasing frameworks for time series data augmentation. To improve the utility, PART-GAN is incorporated with a combination of optimization strategies, e.g., weight pruning and grouping, generator selecting, and denoise mechanisms.
(4) We conduct extensive empirical evaluations using real-world medical time series data, e.g., EEG data and eICU data, to validate the efficacy of PART-GAN. We demonstrate PART-GAN's ability to effectively support augmentation tasks for medical time series data while providing theoretically guaranteed privacy protection and maintaining a reasonable (single-digit) privacy budget.

The next section describes the preliminary concepts. Sections 2 and 3 explain the system design and our optimization strategies. Section 4 describes our experimental results. Section 5 concludes the work as a whole.

2 PART-GAN Schemes

This section describes the idea of PART-GAN, a generic scheme for differentially private releasing of synthetic time series data using CT-GAN for time series mimicking.

2.1 CT-GAN Architecture

In this section, the CT-GAN framework, improved from ACGAN [16] and Conditional GAN (CGAN) [15], is proposed to model the temporal irregularity of the incomplete time series and guide the generation using a given classification. CT-GAN integrates timestamps as the conditional class (similar to [18]) using CGAN to generate irregular time-series sequence while composing with an auxiliary decoder to reconstruct class labels using ACGAN to guide the generation under a given label. We feed side information (timestamps) to the discriminator and the reconstructing side information (classification label). The discriminator D is modified to comprise an auxiliary decoder network that yields the class label for the training samples or a subsection of the latent variables from which the samples are generated. As shown in Fig. 1, the CT-GAN framework is composed of a generator network G to yield synthetic examples with the same distribution as that of real examples; and a discriminator neural network D to distinguish between real and synthetic samples.

In the CT-GAN, each generated sample has a corresponding class label $l \sim L$ and temporal timestamp $t \sim T$ as two kinds of conditional information, in addition to the noise $z \sim Z$. Z is a noise space used to seed the generative model. T and L is the space of timestamps and labels used to condition G and D. Let G generate a segment $x = <x_1, \ldots x_n>$ with n timestamps, similar with the size of x, $t = <t_1, \ldots t_n>$ is a sorted vector of timestamps sampled from T. G uses both x, l and t to mimic time series $x_{fake} = G(t, z, l)$. x represents a time series output from G or input to D. Samples of $z \in Z$ are sampled from a noise distribution $p_z(z)$. In our experiments, p_z is a simple Gaussian noise distribution with a mean equal to 0, and the standard deviation equals 1. Based on the time series in the training data and their associated conditional data, we can define a data distribution $p_{data}(x, t, l)$. The D generates both a probability distribution over fake or real and a probability distribution over the class labels, namely $P(S|X), P(C|X) = D(X)$. The objective function is composed of three parts: the log-likelihood of the corresponding fake or real source (LS), the log-likelihood of the corresponding class (LC), and the loss of temporal conditional (LT).

$$\begin{aligned} L_S &= E[logP(S = real|X_{real})] - E[logP(S = fake|X_{fake})] \\ L_C &= E[logP(C = l|X_{real})] - E[logP(C = l|X_{fake})] \\ L_T &= E_{x \sim p_{data}(x)}[logD(x|t)] + E_{z \sim P_z(z)}[log(1 - D(G(z|t)))] \end{aligned} \tag{1}$$

2.2 (ϵ, δ)-Differential Privacy

One variant of differential privacy is used in this work, i.e. (ϵ, δ)-differential privacy [5], which allows for the possibility that plain ϵ-differential privacy is broken with probability δ[1]. Formally,

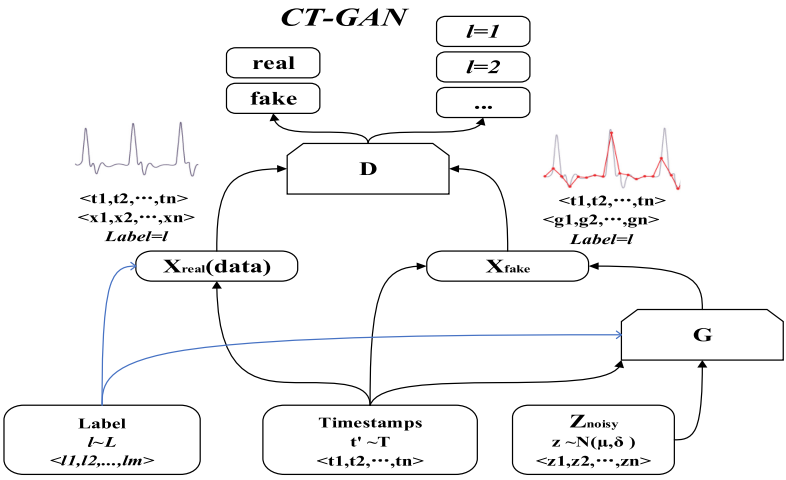


Fig. 1. Structure of CT-GAN.

Definition 1. *(ϵ, δ)-Differential privacy* [4]

Let $\epsilon > 0$ be a constant. A randomized mechanism $\mathcal{M}(\Delta)$ satisfies (ϵ, δ)-differential privacy if, for any two adjacent inputs d_1 and d_2 and for any subset of outputs S ($S \subset$ Range (F)), there is:

$$Pr[\mathcal{M}(d_1) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(d_2) \in S] + \delta \tag{2}$$

The parameter $\epsilon > 0$ is called the privacy budget and it allows users to control the level of privacy. δ evaluates the violation of the ϵ-differential privacy, which allows for the possibility that plain ϵ-differential privacy is broken with probability δ. Given the deterministic function, differential privacy can be satisfied by adding random noise into the output of the deterministic function, where the scale of noise is decided by the sensitivity of Δf. The sensitivity reveals the maximum change in the query answers due to the change of a single database entry.

2.3 PART-GAN Scheme

This work aims to preserve privacy during the training procedure instead of adding random noise to the final parameters. Since GANs consist of both a generator G and a discriminator D, the direct approach is to add noise to the training procedures of both G and D. However, the minimax game formulation makes it hard to accurately obtain an evaluation of the privacy loss, causing large distortion in the generative models. As illustrated in Fig. 1, only the discriminator D directly accesses the real data. Therefore, it is reasonable to inject random noise only in training discriminator D, namely perturbing the training discriminator D is sufficient to protect privacy [1]. Besides, the architecture and number of parameters of discriminator D are small and straightforward. Thus, it is feasible to tightly estimate the privacy loss of perturbation in training D.

Therefore, we add Gaussian noise into the training of the discriminator to guarantee differential privacy. The computation of parameters of the generator can be considered as a post-processing procedure, with the differentially private parameters of the discriminator as the only input. Based on the post-processing property of differential privacy, the calculation of parameters, and the generation of synthetic data of the generator both satisfy differential privacy. Thus, even if the observer has obtained the generator itself, it is hard for the observer to invade the training data's privacy. In our approach, we trained two DNNs against each other to generate realistically simulated synthetic participant samples. Based on the commonly-adopted approaches applying differential privacy to DNNs, PART-GAN satisfies differential privacy by adding random noise to the SGD [1,21]. We then implement differential privacy training D by injecting random noise in the optimization procedure, i.e., SGD, when updating D.

Algorithm 1: PART-GAN Algorithm

Output: Differentially private generator G.
Initialize discriminator and generator parameters θ_d^0, θ_g^0;
while θ_g *has not converge &* $\delta > \delta_0$ **do**
  for *t=1 to* n_g **do**
    //Single-shot Pruning Pre-Processing;
    for *i=1 to* n_d **do**
      //Parameter Grouping Optimization;
      Sampling $\{x^{(j)}\}_{j=1}^{m} \backsim p_{data}$, $\{z^{(j)}\}_{j=1}^{m} \backsim p_z$, $\{t^{(j)}\}_{j=1}^{m} \backsim T$,$\eta \backsim [0,1]$;
      $\{\widehat{x}^{(j)}\}_{j=1}^{m} \leftarrow \eta x + (1-\eta)G(\{z^{(j)}\}_{j=1}^{m})$;
      Gradient $g_w(\widehat{x}^{(j)}, z^{(j)}) \leftarrow \nabla_w[f_w(\widehat{x}^{(j)}) - f_w(g_\theta(z^{(j)}))]$;
      $\overline{g}_w = \frac{1}{m}\sum_{j=1}^{m} g_w(\widehat{x}^{(j)}, z^{(j)})$;
      $g'_w \leftarrow Clipping(\overline{g_w}, C) + \mathcal{N}(0, (\Delta f)^2 C^2 \mathcal{I})$;
    Updating privacy accountant $\mathcal{A}$ with (σ, m_d, m);
    Updating discriminator w with (w, g'_w, α_d);
  Sampling $\{z^{(j)}\}_{j=1}^{m} \backsim p_z$;
  $\overline{g}_\theta = \frac{1}{m}\sum_{j=1}^{m} \nabla_\theta f_w(g_\theta(z^{(j)}))$;
  Updating the generator θ with $(\theta, g_\theta, \alpha_g)$;
  //Generator Selecting Optimization;
  Cumulative privacy loss $\delta \leftarrow$ query $\mathcal{A}$ with ϵ_0;

We introduce the paradigm of PART-GAN in Algorithm 1. The inputs are as follows: α_d, α_g, the learning rate of the discriminator and generator; C, gradient clipping bound constant. n, the total number of training data points in each discriminator iteration; σ_n, noise scale; n_g, number of iterations of the generator; n_d, number of discriminator's iterations per generator iteration; m_d, number of discriminator's parameters; m, batch size for training GAN; (ϵ_0, δ_0), the overall privacy target; k the number of parameter groups. As illustrated, at every stage of the SGD, when the gradient $\nabla_\theta L(\theta; x_i)$ is calculated for a random subset of examples, the l_2-norm of gradients are clipped by a threshold C at first to bound the sensitivity by a constant C. Here, f_w is the function used as the loss function. We then obtain the average of these gradient values, followed by injecting random noise sampled from a Gaussian distribution that

has a variance proportional to the gradient clipping to protect the privacy. Note that a privacy accountant A is adopted in this work, similar to [1], to track the cumulative privacy loss in each step. This progress is iterated until convergence or exceeding the privacy budget. Our empirical results illustrate that directly injecting noise into the training procedure of D encounters several challenges: (1) the utility of generated examples is limited and unrealistic; and (2) the differentially private GAN converges slower than the ordinary GAN, giving rise to unnecessary privacy loss. Therefore, we implement three optimization strategies to improve the training stability and utility of differentially private GAN and convergence rate. **(1) Single-Shot Pruning Pre-Processing.** We use a connection sensitivity definition to discover important connections in the neural network. In the training and perturbation, only the top k important connections are selected to be trained and perturbed while the rest will be skipped from these procedures and kept inactive. Based on the connection sensitivity, it is feasible to identify and prune unimportant connections in single-shot before training, resulting in a perturbation-free mask. **(2) Parameter Dynamic Grouping.** We aggregate parameters with small gradient values when their gradient values are similarly small and close, and the parameters have a similar trend of gradient value change, aiming to reduce the scale of noise introduced to satisfy differential privacy. During each training iteration, we first group the parameters into k groups $\{G_j\}_j^k = 1$, each G_j sharing similar clipping bound c_j. We then use the average clipping bounds in G_j to estimate c_j, followed by group-wise clipping and perturbation. **(3) Generator Selecting Optimization.** We save all the generative models, e.g., generator G, across all epochs followed by selecting generators that can produce training samples for the most accurate models in a differential privacy-preserving manner. Instead of directly adding noise to each gradient, we choose appropriate noisy data that was previously published and reuse it if it is close to the real gradient, which we want to perturb. The details for these improvements are illustrated in Sect. 3.

3 Optimizations

3.1 Single-Shot Pruning Optimization

As neural networks are usually overparameterized, comparable performance can be obtained by pruning the neural network into a smaller network while enhancing generalization [3]. Intuitively, the pre-processing goal is to propose an optimization that can build a sparse network while maintaining the accuracy of the network by selectively pruning redundant connections for a given task in a data-dependent way before training. We assume that besides the private datasets D_{pri} to train the model, a small amount of relevant public dataset D_{pub} is available in many settings, e.g., public human face dataset or public EEG or eICU database.

First, a weight-independent criterion is selected to measure and identify important (or sensitive) connections used for pruning. The neural network pruning can be formulated as an optimization problem. Given a dataset $D = \{(x_i, y_i)\}_{i=1}^n$, a desired pruning rate p (i.e., the number of non-zero weights),

auxiliary indicator variables $c \in \{0,1\}^m$ representing the presence of each connection (e.g., the value of c_j indicates whether the connection j is active in the network or pruned), neural network pruning can be written as the following constrained optimization problem:

$$\begin{aligned} \underset{c,w}{Min} L(c \odot w; D) = \underset{c,w}{min} \frac{1}{n} \sum_{i=1}^{n} l(c \odot w; (x_i, y_i)) \\ s.t. w \in R^m, c \in \{0,1\}^m, \|c\|_0 < p \end{aligned} \tag{3}$$

Here, $l(\Delta)$ is the standard loss function (e.g., cross-entropy loss), where $\odot$ is the Hadamard product, w is the set of parameters of the neural network, and m is the total number of parameters. This measure is used to separate the weight of the connection (w) from whether the connection is present or not (c) [11]. Then the importance of each connection can be decided by measuring its effect on the loss function. Specifically, to measure the effect of connection j on the loss, the difference in loss can be evaluated separately on $c_j = 1$ and $c_j = 0$. The effect of removing connection j, denoted by ΔL_j, can be measured by:

$$\Delta L_j(w; D) = L(\mathbf{1} \odot \mathbf{w}; D) - L((\mathbf{1} - \mathbf{v}_j) \odot \mathbf{w}; D) \tag{4}$$

where 1 is the vector of dimension m and v_j is the indicator vector of element j. ΔL_j can be approximated by the derivative of L w.r.t. c_j , denote by $d_j(w; D)$. Consequently, the effect of connection j on the loss can be defined as [10,11]:

$$\begin{aligned} \Delta L_j(\mathbf{w}; D) \approx d_j(\mathbf{w}; D) = \frac{\partial L(\mathbf{c} \odot \mathbf{w}; D)}{\partial c_j}|_{c_j=1} \\ = \lim_{\delta \to 0} \frac{L(\mathbf{c} \odot \mathbf{w}; D) - L((\mathbf{c} - \delta \mathbf{e}_j) \odot \mathbf{w}; D)}{\delta}|_{c_j=1} \end{aligned} \tag{5}$$

As an infinitesimal version of ΔL_j, $\partial L / \partial c_j$ can be figured out on one forward-backward pass using automatic differentiation for all j at once. Note that a higher magnitude of the derivatives essentially means that the connection c_j has a significant positive or negative effect on the loss [11]. Then, connection sensitivity cs for j can be defined as the normalized magnitude of the derivatives:

$$cs_j = \frac{|d_j(w; D)|}{\sum_{k=1}^{m} |d_k(w; D)|} \tag{6}$$

Neural network initialization [11] that guarantees the gradients to be in a reasonable range is adopted to relieve the impact of weights on the derivatives. The single-shot pruning-based optimization algorithm is described in Algorithm 2, in which a relevant public dataset is used to initialize and calculate the connection sensitivity of the model. This optimization is placed before the Algorithm 1 as the pre-processing, where the mask is used for gradient perturbation.

3.2 Parameter Dynamic Grouping

The common approach to satisfy differential privacy for SGD optimization is to directly add random noise to each gradient value. This straightforward way

Algorithm 2: Single-shot pruning-based optimization

Input: Loss function L, relevant public dataset D_{pub}, pruning rate k
Output: Perturbation-free mask

Construct random batch B with size $|B| = K$;
Calculate $\nabla L_B(x_t)$ and V_B; **while** $\|w^*\|_0 \leq k$ **do**
  $w_0 \leftarrow VarianceScalingInitialization$;
  // Sample a mini-batch from D_{pub}
  cs $\leftarrow$ Calculate $cs_j \forall j \in \{1, \cdots, m\}$;
  c $\leftarrow$ top k c_j according to DescendingSorting(**cs**);
Return **c**;

introduces great perturbation, especially for gradients with small values. To this end, grouping optimization [22,24] has been applied to aggregate parameters with small values to reduce the amount of introduced noise.

In this section, we use a dynamic grouping approach that can dynamically aggregate parameters with small gradient values when their gradient values are similarly small and close, and the parameters have a similar trend of gradient value change. According to the post-processing property of the differential privacy definition, no privacy is compromised when using the perturbed gradient values of previous iterations to predict the trend of gradient value change and estimated gradient value at the current iteration.

Formally, given perturbed gradient value set $\{g^a_{i-k}, g^a_{i-k+1}, \cdots, g^a_{i-1}\}$ at k previous iterations for one parameter a, the estimated gradient value gradient for a at a i_{th} iteration is predicted as $\overline{x}^a_i = \sum_{j=i-k}^{i-1} g^a_i/k$. The Pearson Correlation Coefficient is further used to represent the similarity of gradient value change trend for various parameters at the i_{th} iteration, based on the purebred gradient values at k previous iterations. Then, parameters with small predicted gradient value and high similarity will be aggregated together to generate groups. Finally, we treat each group as a single value and then perturb it with a Gaussian mechanism. Specifically, we determine the sum of gradient values for each group at first, then add Gaussian noise to perturb the sum. The average perturbed value is used as the perturbed gradient for each parameter in the group.

As demonstrated by the example in Fig. 2, there are three parameters at the current i^{th} iteration and perturbed gradient values. At least three iterations are used to estimate the gradient value and change trend at current iteration (circle in green colour). Given $\tau_1 = 20, \tau_2 = 0.5, \tau_3 = 10, k = 3$, the value of each circle is the perturbed gradient values and the estimated gradient $(\overline{x}^a_i)$ for three parameters at the current iteration is $(10+12+18)/3 = 13.3$, $(6+10+13)/3 = 9.7$ and $(48 + 62 + 80)/3 = 63.3$, respectively. As $63.3 > 20$, θ_3 is set as a separate group. For parameter θ_1 and θ_2, their Pearson Correlation Coefficient with their k previous perturbed gradient set $[10, 12, 18]$ and $[6, 10, 13]$ is 0.9347. Since $0.9347 > 0.5$, and $13.7 - 9.7$ is smaller than 10, we can group the two parameters together.

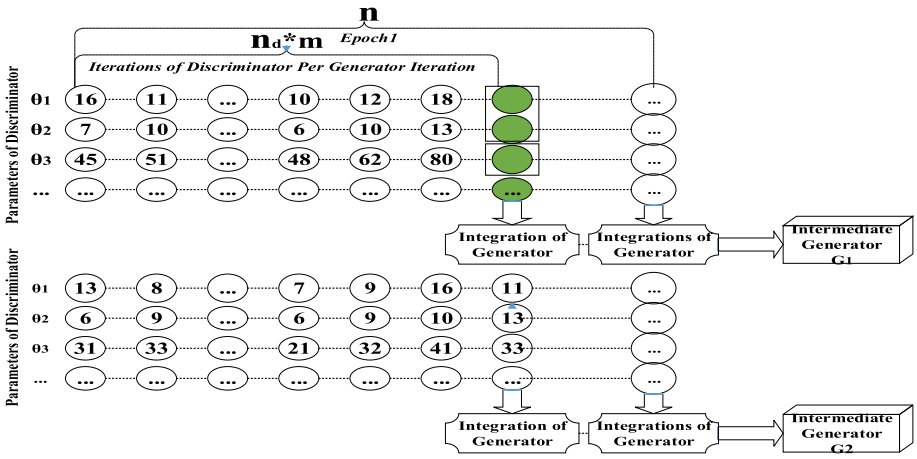


Fig. 2. The training progress of GAN.

3.3 Generator Selecting Optimization

As shown in Fig. 2, all the trained generators across all epochs, e.g., $G1, G2, ...$, are archived as the GAN is trained. We apply targeted learning tasks to evaluate each intermediate generator. Specifically, we produce a batch of synthetic samples using each intermediate generative model and implement some specific machine learning algorithms, e.g., SVM, to build a prediction model based on these synthetic samples. We then test each prediction model using the original training samples to decide the accuracy of each prediction model. Based on the accuracy evaluation, we select these intermediate generators with better resulting accuracy to generalize the final generator.

To protect privacy in the generator selection, we apply the differentially private mechanism to perturb the selecting procedure. Specifically, to select a generator that can produce training samples for the most accurate machine learning models in a differential privacy-preserving manner, we use the standard "Report Noisy Max" subroutine [6]. For example, independent Laplace noise is injected to the accuracy of each prediction model, which is drawn from Laplace distribution $Lap(\Delta f/\epsilon)$ to achieve $(\epsilon, 0)$-differential privacy. Here Δf is the sensitivity for queries.

4 Experimental Results

This section presents the result from empirical evaluations of the proposed PART-GAN over two benchmark datasets (EGG and eICU). These experiments aim to investigate the practical use of privacy-preserving GAN on synthetic data release: (1) Does PART-GAN enable the synthesization of realistic medical datasets, e.g., ICU data, with differential privacy perturbation? (2) Does the generated data retain satisfactory quality and utility for various data analysis

tasks? (3) How do different optimization strategies affect the performance of PART-GAN?

4.1 Datasets and Settings

Datasets. In our experiments, we use two benchmark datasets:

EEG Database. We use the EEG signals from nine subjects provided by the Institute for Knowledge Discover[1]. The dataset contains data about the motor imagery of four different motor imagery tasks: the imagination of movement of the left hand (class 1), right hand (class 2), both feet (class 3), and tongue (class 4).

Philips eICU Database[2]. This dataset consists of about 200,000 patients from 208 care units across the United States, with a total of 224,026,866 entries divided into 33 tables. From this data, we focus on generating the four most frequently recorded, regularly-sampled variables measured by bedside monitors: oxygen saturation measured by pulse oximeter (SpO2), heart rate (HR), respiratory rate (RR) and mean arterial pressure (MAP).

Training and Architecture Setting. We use WGANs to further improve training stability, and train the network according to the setup described in [9], incorporating our perturbation mechanism and optimization strategies. We also scale the penalty term λ by the current critic difference $\tilde{W}(\mathbb{P}_r, \mathbb{P}_\theta)$ and use the one-sided penalty

$$P_1(\mathbb{P}_{\widehat{x}}) = \lambda E_{\widehat{x}\sim\mathbb{P}_{\widehat{x}}}[max(0, (\|\nabla_{\widehat{x}} D(\widehat{x})\|_2 - 1)^2)]$$

The loss function for the critic is

$$L = -\tilde{W}(\mathbb{P}_r, \mathbb{P}_\theta) + max(0, \tilde{W}(\mathbb{P}_r, \mathbb{P}_\theta)) \times P_1(\mathbb{P}_{\widehat{x}})$$

The λ is set at 10 [8]. Each stage is trained for 2000 epochs. Latent variables Z for the generator are sampled from $\mathcal{N}(0, 1)$.

Here, the default parameters and threshold values were set as $\tau_1 = 20, \tau_2 = 0.5, \tau_3 = 10, k = 3, \eta = 0.002$, $\sigma_{EEG} = 0.561, \sigma_{eICU} = 0.598$, $K = 6$. The default values are used if no other values have been specified. Note that the experiments were performed on three classical differential privacy leakage levels, i.e., $(\epsilon, \delta) = \{Strong : (1, 10^{-5}); Weak : (4, 10^{-5}); VeryWeak : (15, 10^{-5})\}$. Each experiment was tested 10 times and the average result reported.

Metrics. The performance of the proposed PART-GAN is evaluated in terms of three different aspects: the data utility-based evaluation of the synthesis mechanism, the privacy-based evaluation of the perturbation mechanism, and the effectiveness of optimization. The detailed evaluation metrics are given as follows:

[1] http://www.bbci.de/competition/iv/#datasets.
[2] https://eicu-crd.mit.edu/.

(1) Data utility-related evaluation. The statistical distributions and cumulative distribution of each attribute are used to compare the statistical similarity between the original data and the corresponding attribute in perturbed/synthesized data. Also, the inception score (IS) [19] is adopted to evaluate the quality of data generated by GAN, $IS(G) = exp(\mathbb{E}_{x\sim G(z)} KL(Pr(y|x)||Pr(y)))$. The x is a sample generated by generator G, $Pr(y|x)$ is the conditional distribution of a pre-trained classifier to predict x's label y. A small entropy of $Pr(y|x)$ reveals that x is similar to a real sample. The marginal distribution of y is $Pr(y) = \int_x Pr(y|x = G(z))dz$. Note that baseline classifiers to predict $Pr(y|x)$ can be trained on an entire training set. The inception score $s(G)$ can describe both the quality and diversity of the synthetic data. However, since the inception score only relies on the final probabilities of the classifier, it is sensitive to noise and is not able to detect mode collapse. Thus, a pre-trained Deep4 model [20] is adopted as a replacement for the inception model. Besides, the correlation structure of attributes is used to compare the statistical similarity between the original data and the corresponding attribute in perturbed/synthesized data.

(2) Privacy-related evaluation. Distance-based metrics are widely used in many data privacy works. Euclidean distance can be used to evaluate how similar generated samples from the PART-GAN (PGS) are to the generated samples from ordinary GAN (OGS). By comparing the distances between OGS and PGS samples, we can investigate the distortion of generated samples from the perturbed generator. Here, we use the minimal distance (MD), i.e., Euclidean distance between a sample s of PGS to the sample closest to s in OGS. Optimally, the distribution of MD between samples of OGS and PGS should be equivalent to the distribution of MDs between only real samples with others than themselves. We calculate the distance after attribute-wise normalization because each attribute contributes equally to the distance after the normalization.

We compare PART-GAN performance with existing ordinary privacy-preserving GAN approaches [7,17,23,24] under differential privacy and ordinary GAN without perturbation using our evaluation metrics. (a) Original database. The baseline method for the data utility-related evaluations is to perform such evaluations on ordinary training samples, denoted by **original**. (b) Ordinary GAN approach without perturbation mechanism. To conduct the privacy-related evaluations, the synthetic samples generated from the ordinary GAN without perturbation mechanisms are used as the baseline datasets, denoted by **GAN**. (c) Regular GAN under differential privacy. To evaluate the improvements of our PART-GAN, the synthetic samples generated from the regular privacy-preserving GAN approaches under differential privacy are used as the baseline datasets, denoted by **DP-GAN** (e.g., commonly-adopted DP-SGD approaches [1,23,24]).

(3) The effectiveness of optimization strategies. We evaluate the impact of multi-folds optimization strategies on PART-GAN performance in terms of allowed iterations and Inception Score before and after applying optimization strategies.

4.2 Evaluation

In this section, a set of quantitative evaluations are conducted to evaluate the performance of PART-GAN, to address the second question: *Does the generated data retain satisfactory quality and utility for various data analysis tasks?*

Table 1. IS of original and synthetic data on MNIST/EEG/eICU dataset

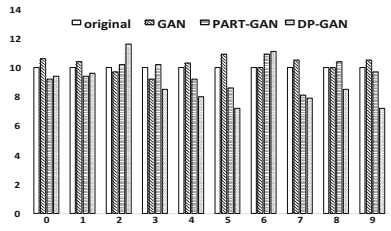
Database	Model	n(10^4)	(ϵ, δ)	IS
MINIST/EEG/eICU	Original	5/200/300	–	$9.87 \pm 0.03/5.27 \pm 0.02/6.56 \pm 0.04$
	GAN	5/200/300	–	$8.98 \pm 0.03/4.01 \pm 0.02/5.42 \pm 0.04$
	DP-GAN	4/180/280	(5/10/15, 10^{-5})	$8.45 \pm 0.03/3.09 \pm 0.02/4.11 \pm 0.04$
	PART-GAN	4/180/280	(5/10/15, 10^{-5})	$8.77 \pm 0.03/3.89 \pm 0.02/5.07 \pm 0.04$

Inception Score Evaluation. Table 1 summarizes the Inception scores of original and synthetic data (generated by GAN with and without perturbation mechanism) for the EEG and eICU datasets. We found that PART-GAN enables us to synthesize data with Inception scores close to the original data, similar to ordinary GAN and better than DP-GAN. For EEG, the difference between the real data and the synthetic data by PART-GAN is less than 1.4, increasing from 3.09 to 3.89 compared to regular DP-GAN. For eICU, the difference between the real data and the synthetic data by PART-GAN is less than 1.5, increasing from 4.11 to 5.07 compared with regular DP-GAN.

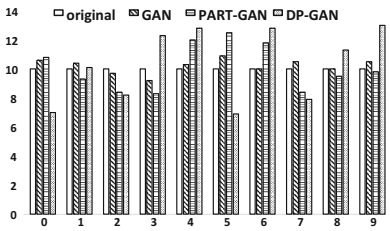
Statistics and Cumulative Distribution Evaluation. The statistical distributions and cumulative distributions of specific sensitive attributes for original and synthetic data are demonstrated in Fig. 3, 4, 5 and 6.

Figures 3(a–b) and 4(a–b) are statistics of the sample label in the MNIST and EEG using ordinary GAN without perturbation, regular DP-GAN and our improved PART-GAN under various privacy-preserving level perturbation. We find that each label's statistics results using PART-GAN with the low-privacy setting are more realistic than regular GAN under the same setting.

Figures 5(a–b) and 6(a–b) show four cumulative distributions of the C3 of EEG attribute of EEG and the MAP attributes of the eICU dataset using ordinary GAN, regular DP-GAN, and PART-GAN under low and high privacy-preserving level perturbation. PART-GAN with the low-privacy setting produces a more realistic cumulative distribution and a wider range of values than others.

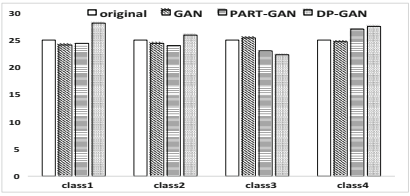


(a) Under high perturbation $\epsilon = 1, \delta \leq 10^{-5}$

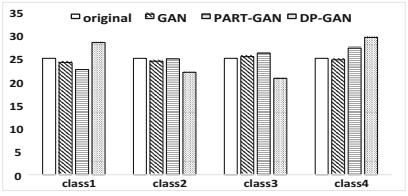


(b) Under low perturbation $\epsilon = 4, \delta \leq 10^{-5}$

Fig. 3. Label distribution of the generated MNIST dataset via ordinary GAN, regular DP-GAN and PART-GAN. PART-GAN with the low-privacy (perturbation) setting are realistic as regular GAN under the same setting for the MNIST generation task.



(a) Under high perturbation $\epsilon = 1, \delta \leq 10^{-5}$



(b) Under low perturbation $\epsilon = 4, \delta \leq 10^{-5}$

Fig. 4. Label distribution of the generated EEG dataset via ordinary GAN, regular DP-GAN and PART-GAN. PART-GAN with the low-privacy (perturbation) setting are realistic as regular GAN under the same setting for the EEG generation task.

In all cases, synthetic tables are statistically similar to the original table. PART-GAN with the high-privacy setting performs better than ordinary GAN. The optimization strategies show a better synthesis performance than both ordinary GAN and improved GAN without optimization.

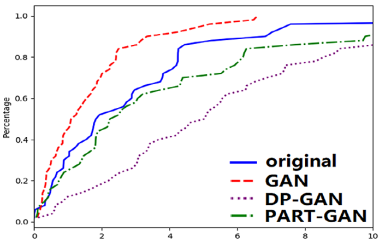
At last, the efficiency of different optimization strategies on PART-GAN's performance is evaluated, to address the third question: *How do different optimization strategies affect the performance of PART-GAN?*

Table 2. Effect on allowed iterations #.

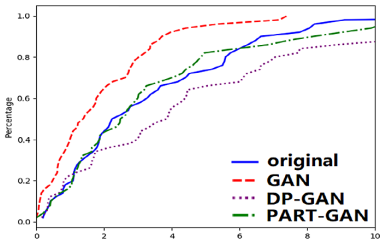
Itereations	# before	# after
MNIST	590	820
EGG	3300	4500
eICU	15000	22000

We first evaluate the optimization strategies on the number of allowed iterations given the same privacy constraints. We illustrate the maximum number

of allowed iterations before and after applying optimization strategies, under the same privacy constraint. The number of allowed iterations is significantly increased by optimization strategies (from 3300 to 4500 for EGG and 15000 to 22000 for eICU), resulting in improved utility in the generators. Then, the effect of optimization strategies on the performance of PART-GAN is demonstrated in Table 2 for the EGG (labeled database) and eICU (unlabeled database), taking the Inception score as an example. It is clear that the performance of PART-GAN can be significantly improved by combining multi-fold optimization strategies.

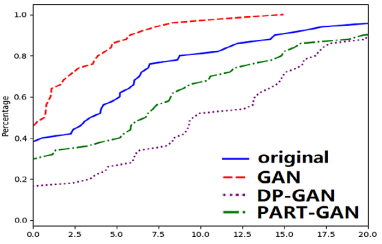


(a) Under high perturbation $\epsilon = 1, \delta \leq 10^{-5}$

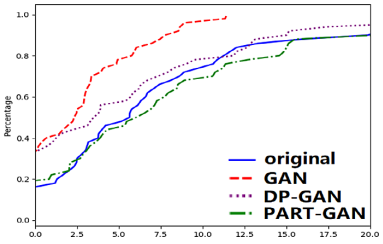


(b) Under low perturbation $\epsilon = 4, \delta \leq 10^{-5}$

Fig. 5. Cumulative distribution of attribute C3 for the generated EGG dataset via ordinary GAN, regular DP-GAN and PART-GAN. PART-GAN under low-privacy (perturbation) generates a more realistic cumulative distribution and a wider range of values than others for EGG generation task.



(a) Under high perturbation $\epsilon = 1, \delta \leq 10^{-5}$



(b) Under low perturbation $\epsilon = 4, \delta \leq 10^{-5}$

Fig. 6. Cumulative distribution of attribute MAP for the eICU dataset by ordinary GAN, regular DP-GAN and PART-GAN. PART-GAN under low-privacy (perturbation) generates a more realistic cumulative distribution and a wider range of values than others for eICU generation task.

5 Conclusion

In this paper, we proposed PART-GAN, a generic scheme of publishing to synthesize time-series data in a privacy-preserving manner. Instead of releasing sanitized datasets, PART-GAN releases differentially private generative models, which can be used by analysts to synthesize an unlimited amount of data for arbitrary analysis tasks. To achieve this, PART-GAN uses a CT-GAN model for the generation of time series data with irregular sampling and under the guidance of a given classifier label. Then PART-GAN integrates the CT-GAN framework with differential privacy mechanisms and employs a suite of optimization strategies to address the utility and training stability challenges. To our knowledge, our method is the first attempt to synthesize realistic real-world time-series databases using GAN techniques in a practical differential privacy-preserving manner, combining the utility and stability optimizations of privacy-preserving GAN into a single task.

References

1. Abadi, M., et al.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 308–318. ACM (2016)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN. arXiv preprint arXiv:1701.07875 (2017)
3. Arora, S., Ge, R., Neyshabur, B., Zhang, Y.: Stronger generalization bounds for deep nets via a compression approach. arXiv preprint arXiv:1802.05296 (2018)
4. Cynthia, D.: Differential privacy. In: Automata, Languages and Programming, pp. 1–12 (2006)
5. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: privacy via distributed noise generation. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 486–503. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_29
6. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Found. Trends® Theor. Comput. Sci. **9**(3–4), 211–407 (2014)
7. Esteban, C., Hyland, S.L., Rätsch, G.: Real-valued (medical) time series generation with recurrent conditional GANs. arXiv preprint arXiv:1706.02633 (2017)
8. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of Wasserstein GANs. In: Advances in Neural Information Processing Systems, pp. 5767–5777 (2017)
9. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
10. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1885–1894. JMLR. org (2017)
11. Lee, N., Ajanthan, T., Torr, P.H.: SNIP: single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:1810.02340 (2018)
12. Li, Y., Swersky, K., Zemel, R.: Generative moment matching networks. In: International Conference on Machine Learning, pp. 1718–1727 (2015)

13. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders. arXiv preprint arXiv:1511.05644 (2015)
14. Mescheder, L., Nowozin, S., Geiger, A.: Adversarial variational bayes: unifying variational autoencoders and generative adversarial networks. arXiv preprint arXiv:1701.04722 (2017)
15. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
16. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 2642–2651. JMLR. org (2017)
17. Park, N., Mohammadi, M., Gorde, K., Jajodia, S., Park, H., Kim, Y.: Data synthesis based on generative adversarial networks. Proc. VLDB Endow. **11**(10), 1071–1083 (2018)
18. Ramponi, G., Protopapas, P., Brambilla, M., Janssen, R.: T-CGAN: conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. arXiv preprint arXiv:1811.08295 (2018)
19. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: Advances in Neural Information Processing Systems, pp. 2234–2242 (2016)
20. Schirrmeister, R.T., et al.: Deep learning with convolutional neural networks for eeg decoding and visualization. Hum. Brain Mapp. **38**(11), 5391–5420 (2017)
21. Song, S., Chaudhuri, K., Sarwate, A.D.: Stochastic gradient descent with differentially private updates. In: 2013 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 245–248. IEEE (2013)
22. Wang, Q., Zhang, Y., Lu, X., Wang, Z., Qin, Z., Ren, K.: Real-time and spatio-temporal crowd-sourced social network data publishing with differential privacy. IEEE Trans. Dependable Secure Comput. **15**, 591–606 (2016)
23. Xie, L., Lin, K., Wang, S., Wang, F., Zhou, J.: Differentially private generative adversarial network. arXiv preprint arXiv:1802.06739 (2018)
24. Zhang, X., Ji, S., Wang, T.: Differentially private releasing via deep generative model. arXiv preprint arXiv:1801.01594 (2018)