Intelligent Decision Technologies -1 (2021) 1–29 1
DOI 10.3233/IDT-210195
IOS Press

Differentially private synthetic mixed-type data generation for unsupervised learning

Uthaipon Tao Tantipongpipat[a], Chris Waites[b], Digvijay Boob[c], Amaresh Ankit Siva[d] and Rachel Cummings[e,*]
[a]
[b]*Stanford University, Stanford, CA, USA*
[c]*Southern Methodist University, Dallas, TX, USA*
[d]*Amazon*
[e]*Columbia University, New York, NY, USA*

Abstract. We introduce the DP-auto-GAN framework for synthetic data generation, which combines the low dimensional representation of autoencoders with the flexibility of Generative Adversarial Networks (GANs). This framework can be used to take in raw sensitive data and privately train a model for generating synthetic data that will satisfy similar statistical properties as the original data. This learned model can generate an arbitrary amount of synthetic data, which can then be freely shared due to the post-processing guarantee of differential privacy. Our framework is applicable to unlabeled *mixed-type data*, that may include binary, categorical, and real-valued data. We implement this framework on both binary data (MIMIC-III) and mixed-type data (ADULT), and compare its performance with existing private algorithms on metrics in unsupervised settings. We also introduce a new quantitative metric able to detect diversity, or lack thereof, of synthetic data.

Keywords: Differential privacy, synthetic data generation, generative adversarial networks, mixed-type data

1. Introduction

As data storage and analysis are becoming more cost effective, and data become more complex and unstructured, there is a growing need for sharing large datasets for research and learning purposes. This is in stark contrast to the previous statistical model where a data curator would hold datasets and answer specific queries from (potentially external) analysts. Sharing entire datasets allows analysts the freedom to perform their analyses in-house with their own devices and toolkits, without having to pre-specify the analyses they wish to perform. However, datasets are often proprietary or sensitive, and they cannot be shared directly. This motivates the need for *synthetic data generation*, where a new dataset is created that shares the same statistical properties as the original data. These data may not be of a single type: all binary, all categorial, or all real-valued; instead they may be of *mixed-types*, containing data of multiple types in a single dataset. These data may also be unlabeled, requiring techniques for *unsupervised learning*, which is typically a more challenging task than *supervised* learning when data are labeled.

Privacy challenges naturally arise when sharing highly sensitive datasets about individuals. Ad hoc anonymization techniques have repeatedly led to severe privacy violations when sharing "anonymized" datasets. Notable examples include the Netflix Challenge [1], AOL Search Logs [2], and Massachusetts State Health data [3], where linkage attacks to publicly available auxiliary datasets were used to reidentify individuals in the dataset. Even deep learning models have been shown to inadvertently memoize sensitive personal information such as Social Security Numbers during training [4].

*Corresponding author: Rachel Cummings, Columbia University, 500 W 120th St., New York, NY 10027, USA. Tel.: +1 212 854 2942; E-mail: rac2239@columbia.edu.

Differential privacy (DP) [5] (formally defined in Section 2) has become the de facto gold standard of privacy in the computer science literature. Informally, it bounds the extent to which an algorithm depends on a single datapoint in its training set. The guarantee ensures that any differentially privately learned models do not overfit to individuals in the database, and therefore cannot reveal sensitive information about individuals. It is an information theoretic notion that does not rely on any assumptions of an adversary's computational power or auxiliary knowledge. Furthermore, it has been shown empirically that training machine learning models with differential privacy protects against membership inference and model inversion attacks [4,6]. Differentially private algorithms have been deployed at large scale in practice by organizations such as Apple, Google, Microsoft, Uber, and the U.S. Census Bureau.

Much of the prior work on differentially private synthetic data generation has been either theoretical algorithms for highly structured classes of queries [7,8] or based on deep generative models such as Generative Adversarial Networks (GANs) or autoencoders. These architectures have been primarily designed for either all-binary or all-real-valued datasets, and have focused on the *supervised* setting.

In this work we introduce the *DP-auto-GAN* framework, which combines the low dimensional representation of autoencoders with the flexibility of GANs. This framework can be used to take in raw sensitive data, and privately train a model for generating synthetic data that satisfies similar statistical properties as the original data. This learned model can be used to generate arbitrary amounts of publicly available synthetic data, which can then be freely shared due to the post-processing guarantees of differential privacy. We implement this framework on both unlabeled binary data (for comparison with previous work) and unlabeled mixed-type data. We also introduce new metrics for evaluating the quality of synthetic mixed-type data in unsupervised settings, and empirically evaluate the performance of our algorithm according to these metrics on two datasets.

1.1. Our contributions

In this work, we provide two main contributions: a new algorithmic framework for privately generating synthetic data, and empirical evaluations of our algorithmic framework showing improvements over prior work. Along the way, we also develop a novel privacy composition method with tighter guarantees, and we generalize previous metrics for evaluating the quality of synthetic datasets to the unsupervised mixed-type data setting. Both of these contributions may be of independent interest.

Algorithmic framework. We propose a new data generation architecture which combines the versatility of an autoencoder [9] with the recent success of GANs on complex data. Our model extends previous autoencoder-based DP data generation [10,11] by removing an assumption that the distribution of the latent space follows a Gaussian mixture distribution. Instead, we incorporate GANs into the autoencoder framework so that the generator must learn the true latent distribution against the discriminator. We describe the composition analysis of differential privacy when the training consists of optimizing both autoencoders and GANs (with different noise parameters).

Empirical results. We empirically evaluate the performance of our algorithmic framework on the MIMIC-III medical dataset [12] and UCI ADULT Census dataset [13], and compare against previous approaches in the literature [10,14–16]. Our experiments show that our algorithms perform better and obtain substantially improved ϵ values of $\epsilon \approx 1$, compared to $\epsilon \approx 200$ in prior work [15]. The performance of our algorithm remains high along a variety of quantitative and qualitative metrics, even for small values of ϵ, corresponding to strong privacy guarantees. Our code is publicly available for future use and research.

1.2. Related work on differentially private data generation

Early work on differentially private synthetic data generation was focused primarily on theoretical algorithms for solving the *query release problem* of privately and accurately answering a large class of pre-specified queries on a given database. It was discovered that generating synthetic data on which the queries could be evaluated allowed for better privacy composition than simply answering all the queries directly [7,8,17,18]. Bayesian inference has also been used for differentially private data generation [19,20] by estimating the correlation between features. See [21] for a survey of techniques used in private synthetic data generation.

More recently, [22] introduced a framework for training deep learning models with differential privacy, which involved adding Gaussian noise to a clipped (norm-bounded) gradient in each training step. [22] also introduced the *moment accountant* privacy analysis, which provided a tighter Gaussian-based privacy composition and allowed for significant improvements in

Table 1

Algorithmic frameworks for differentially private synthetic data generation. Our new algorithmic framework (in **bold**) is the first to combine both DP GANs and autoencoders into one framework by using GANs to learn a generative model in the latent space

Types	Algorithmic framework: Architecture	Algorithmic framework: Variants
Deep generative models	DPGAN [22]	PATEGAN [41]
		DP Wasserstein GAN [36]
		DP Conditional GAN [38]
		Gumbel-softmax for categorical data [14]
	Autoencoder	DP-VAE (standard Gaussian as a generative model in latent space) [11,16]
		RBM generative models in latent space [16]
		Mixture of Gaussian model in latent space [10]
	Autoencoder and DPGAN **(ours)**	
Other models	SmallDB [7], PMW [8], MWEM [17], DualQuery [18], DataSynthesizer [20], PrivBayes [19]	

accuracy. It was later defined in terms of *Renyi Differential Privacy (RDP)* [23], which is a slight variant of differential privacy designed for easy composition. Much of the work that followed used deep generative models, and can be broadly categorized into two types: autoencoder-based and GAN-based. Our algorithmic framework is the first to combine both DP GANs and autoencoders.

Differentially private autoencoder-based models. A variational autoencoder (VaE) [9] is a generative model that compresses high-dimensional data to a smaller space called *latent space*. The compression is commonly achieved through deep models and can be differentially privately trained [11,16]. VaE makes the (often unrealistic) assumption that the *latent distribution* is Gaussian. Acs et al. [16] uses Restricted Boltzmann machine (RBM) to learn the latent Gaussian distribution, and Abay et al. [10] uses expectation maximization to learn a Gaussian mixture. Our work extends this line of work by additionally incorporating the generative model GANs which have also been shown to be successful in learning latent distributions.

Differentially private GANs. GANs are generative models proposed by Goodfellow et al. [24] that have been shown success in generating several different types of data [25–33]. As with other deep models, GANs can be trained privately using the aforementioned private stochastic gradient descent (formally introduced in Section 2.1). In this work, we focus on and compare to previous works where DP have been applied.

Variants of DP GANs have been used for synthetic data generation, including the Wasserstein GAN (WGAN) [34,35] and DP-WGAN [6,36] that use a Wasserstein-distance-based loss function in training [6,34–36]; the conditional GAN (CGAN) [37] and DP-CGAN [38] that operate in a supervised (labeled) setting and use labels as auxiliary information in training; and Private Aggregation of Teacher Ensembles (PATE) [39,40] for the semi-supervised setting of multi-label classification when some unlabelled public data are available (or PATEGAN [41] when no public data are available). Our work focuses on the unsupervised setting where data are unlabeled, and no (relevant) labeled public data are available.

These existing works on differentially private synthetic data generation are summarized in Table 1.

Differentially private generation of mixed-type data. Next we describe the three most relevant recent works on privately generating synthetic mixed-type data. [10] considers the problem of generating mixed-type labeled data with k possible labels. Their algorithm, DP-SYN, partitions the dataset into k sets based on the labels and trains a DP autoencoder on each partition. Then the DP expectation maximization (DP-EM) algorithm of [42] is used to learn the distribution in the latent space of encoded data of the given label-class. The main workhorse, DM-EM algorithm, is designed and analyzed for Gaussian mixture models and more general factor analysis models. [11] works in the same setting, but replaces the DP autoencoder and DP-EM with a DP variational autoencoder (DP-VAE). Their algorithm assumes that the mapping from real data to the Gaussian distribution can be efficiently learned by the encoder.

Finally, [14] uses a Wasserstein GAN (WGAN) to generate differentially private mixed-type synthetic data, which uses a Wasserstein-distance-based loss function in training. Their algorithmic framework privatizes the WGAN using DP-SGD, similar to previous approaches for image datasets [15,43]. The methodology of [14] for generating mixed-type synthetic data involves two main ingredients: changing discrete (categorical) data to binary data using one-hot encoding, and adding an output softmax layer to the WGAN generator for every discrete variable.

Our framework is distinct from these three approaches. We use a differentially private autoencoder

Table 2
Summary of evaluation metrics in DP synthetic data generation literature. We list applicability of each metric to each of the data type. Parts in **bold** are **our new contributions**. Evaluation methods with asterisk * are predictive-model-specific, and their applicability therefore depends on types of data that the chosen predictive model is appropriate for. Methods with asterisks ** are equipped with any distributional distance of choice such as Wasserstein distance or total variation distance

Types	Evaluation methods	Data types		
		Binary	Categorical	Regression
Supervised	Label prediction* [10,11,14]	Yes	Yes	Yes
	Predictive model ranking* [41]	Yes	Yes	Yes
Unsupervised, prediction-based	Dimension-wise prediction plot*	Yes ([47], ours)	**Yes**	**Yes**
Unsupervised, distribution-based	Dimension-wise probability plot [47]	Yes	No	No
	2, 3-way feature marginal, total variation distance [10,44]**	Yes	Yes	Yes
	1-way feature marginal, diversity measure (μ-smooth KL divergence) **	**Yes**	**Yes**	**Yes**
Unsupervised, qualitative	1-way feature marginal (histogram) (e.g. in [53,54] and in the implementation of [14])	Yes	Yes	Yes
	2-way PCA marginal (data visualization)	**Yes**	**Yes**	**Yes**

which, unlike DP-VAE of [11], does not require mapping data to a Gaussian distribution. This allows us to reduce the dimension of the problem handled by the WGAN, hence escaping the issues of high-dimensionality from the one-hot encoding of [14]. We also use DP-GAN, replacing DP-EM in [10], to learn more complex distributions in the latent or encoded space.

NIST differential privacy synthetic data challenge. The National Institute of Standards and Technology (NIST) recently hosted a challenge to find methods for privately generating synthetic mixed-type data [44], using excerpts from the Integrated Public Use Microdata Sample (IPUMS) of the 1940 U.S. Census Data as training and test datasets. Four of the winning solutions have been made publicly available with open-source code [45]. However, all of these approaches are highly tailored to the specific datasets and evaluation metrics used in the challenge, including specialized data pre-processing methods and hard-coding details of the dataset in the algorithm. As a result, they do not provide general-purpose methods for differentially private synthetic data generation, and it would be inappropriate–if not impossible – to use any of these algorithms as baseline for other datasets such as ones we consider in this paper.

Evaluation metrics for synthetic data. Various evaluation metrics have been considered in the literature to quantify the quality of the synthetic data (see Charest [46] for a survey). The metrics can be broadly categorized into two groups: *supervised* and *unsupervised*. Supervised evaluation metrics are used when there are clear distinctions between features and labels of the dataset, e.g., for healthcare applications, a person's disease status is a natural label. In these settings, a predictive model is typically trained on the synthetic data, and its accuracy is measured with respect to the real (test) dataset. Unsupervised evaluation metrics are used when no feature of the data can be decisively termed as a label. Recently proposed metrics include *dimension-wise probability* for binary data [47], which compares the marginal distribution of real and synthetic data on each individual feature, and *dimension-wise prediction* which measures how closely synthetic data captures relationships between features in the real data. This metric was proposed for binary data, and we extend it here to mixed-type data. Recently, NIST [44] used a 3-way marginal evaluation metric which used three random features of the real and synthetic datasets to compute the total variation distance as a statistical score. See Appendix 4 for more details on both categories of metrics, including Table 2 which summarizes the metrics' applicability to various data types.

2. Preliminaries on differential privacy

In the setting of differential privacy, a dataset X consists of m individuals' sensitive information, and two datasets are neighbors if one can be obtained from the other by the addition or deletion of one datapoint. Differential privacy requires that an algorithm produce similar outputs on neighboring datasets, thus ensuring that the output does not overfit to its input dataset, and that the algorithm learns from the population but not from the individuals.

Definition 1 (Differential privacy [5]). For ϵ, $\delta > 0$, an algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private if for any pair of neighboring databases X, X' and any subset S

of possible outputs produced by $\mathcal{M}$,

$$\Pr[\mathcal{M}(X) \in S] \leqslant e^{\epsilon} \cdot \Pr[\mathcal{M}(X') \in S] + \delta.$$

A smaller value of ϵ implies stronger privacy guarantees (as the constraint above binds more tightly), but usually corresponds with decreased accuracy, relative to non-private algorithms or the same algorithm run with a larger value of ϵ. Differential privacy is typically achieved by adding random noise that scales with the *sensitivity* of the computation being performed, which is the maximum change in the output value that can be caused by changing a single entry. Differential privacy has strong *composition guarantees*, meaning that the privacy parameters degrade gracefully as additional algorithms are run on the same dataset. It also has a *post-processing* guarantee, meaning that any function of a differentially private output will retain the same privacy guarantees.

2.1. *Differentially private stochastic gradient descent (DP-SGD)*

Training deep learning models reduces to minimizing some (empirical) loss function $f(X;\theta) := \frac{1}{m}\sum_{i=1}^{m} f(x_i;\theta)$ on a dataset $X = \{x_i \in \mathbb{R}^n\}_{i=1}^{m}$. Typically f is a nonconvex function, and a common method to minimize f is by iteratively performing stochastic gradient descent (SGD) on a batch B of sampled data points:

$$B \leftarrow \textsc{BatchSample}(X)$$
$$\theta \leftarrow \theta - \eta \cdot \frac{1}{|B|}\sum\nolimits_{i\in B} \nabla_\theta f(x_i,\theta) \quad (1)$$

The size of B is typically fixed as a moderate number to ensure quick computation of gradient, while maintaining that $\frac{1}{|B|}\sum_{i\in B}\nabla f(x_i,\theta)$ is a good estimate of true gradient $\nabla_\theta f(X;\theta)$.

To make SGD private, Abadi et al. [22] proposed to first clip the gradient of each sample to ensure the ℓ_2-norm is at most C:

$$\textsc{Clip}(x, C) := x \cdot \min\left(1, C/||x||_2\right).$$

Then a multivariate Gaussian noise parametrized by noise multiplier ψ is added before taking an average across the batch, leading to noisy-clipped-averaged gradient estimate g:

$$g \leftarrow \frac{1}{|B|}\Big(\sum\nolimits_{i\in B}\textsc{Clip}(\nabla_\theta f(x_i,\theta), C) + \mathcal{N}(0, C^2\psi^2 I)\Big).$$

The quantity g is now private and can be used for the descent step $\theta \leftarrow \theta - \eta \cdot g$ in place of Eq. (1).

Performance Improvements. In general, the descent step can be performed using other optimization methods – such as Adam or RMSProp – in a private manner, by replacing the gradient value with g in each step. Also, one does not need to clip the individual gradients, but can instead clip the gradient of a group of datapoints, called a *microbatch* [48]. Mathematically, the batch B is partitioned into microbatches $B_1, \ldots, B_k$ each of size r, and the gradient clipping is performed on the average of each microbatch:

$$g \leftarrow \frac{1}{k}\Big(\sum\nolimits_{i=1}^{k}\textsc{Clip}(\nabla_\theta f(X_{B_i},\theta), C) + \mathcal{N}(0, C^2\psi^2 I)\Big).$$

Standard DP-SGD corresponds to setting $r = 1$, but setting higher values of r (while holding $|B|$ fixed) significantly decreases the runtime and reduces the accuracy, and does not impact privacy significantly for large dataset. Other clipping strategies have also been suggested. We refer the interested reader to [48] for more details of clipping and other optimization strategies.

The improved moment accountant privacy analysis by [22] (which has been implemented in Google [49] and is widely used in practice) obtains a tighter privacy bound when data are subsampled, as in SGD. This analysis requires independently sampling each datapoint with a fixed probability q in each step.

The DP-SGD framework (Algorithm 1) is generically applicable to private non-convex optimization. In our proposed model, we use this framework to train the autoencoder and GAN.

Algorithm 1: DP-SGD (one iteration step)

```
parameter input: Dataset X = {x_i}_{i=1}^m, deep learning model parameter θ, learning rate η, loss function f, optimization method OPTIM, batch sampling rate q (for the batch expectation size b = qm), clipping norm C, noise multiplier ψ, microbatch size r
goal: differentially privately train one step of the model parametrized by θ with OPTIM
procedure DP-SGD
    procedure SAMPLEBATCH(X, q)
        𝓑 ← {}
        for i = 1 ... m do
            Add x_i to 𝓑 with probability q
        Return 𝓑
    Partition 𝓑 into B_1, ..., B_k each of size r (ignoring the dividend)
    k̂ ← qm/r                                        ▷ an estimate of k
    g ← (1/k̂) ( Σ_{i=1}^k CLIP(∇_θ f(X_{B_i}, θ), C) + 𝒩(0, C²ψ²I) )
    θ ← OPTIM(θ, g, η)
```

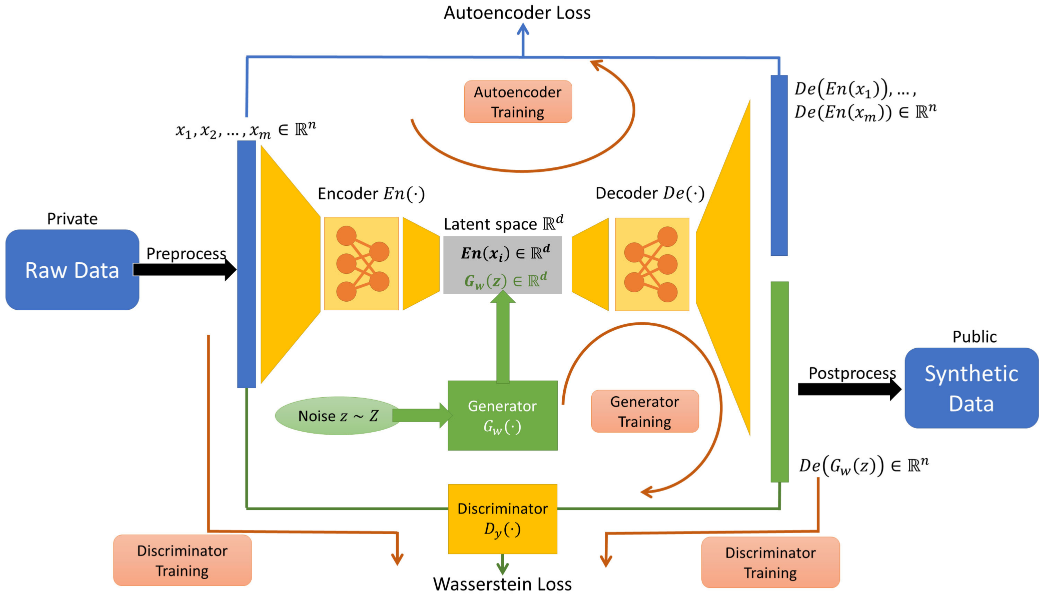


Fig. 1. The summary of our DP-auto-GAN algorithmic framework. Pre- and post-processing (in black) are assumed to be public knowledge. Generator (in green) is trained without noise, whereas encoder, decoder, and discriminator (in yellow) are trained with noise. The four red arrows indicate how data are forwarded for autoencoder, generator, and discriminator training. After training, the generator and decoder are released to the public to generate synthetic data.

2.2. *Renyi differential privacy accountant*

A variant notion of differential privacy, known as *Renyi Differential Privacy (RDP)* [23], is often used to analyze privacy for DP-SGD. A randomized mechanism $\mathcal{M}$ is (α, ϵ)-RDP if for all neighboring databases X, X' that differ in at most one entry,

$$RDP(\alpha) := D_\alpha(\mathcal{M}(X)||\mathcal{M}(X')) \leqslant \epsilon,$$

where $D_\alpha(P||Q) := \frac{1}{\alpha-1} \log \mathbb{E}_{x\sim X} \left(\frac{P(x)}{Q(x)}\right)^\alpha$ is the *Renyi divergence* or *Renyi entropy* of order α between two distributions P and Q. Renyi divergence is better tailored to tightly capture the privacy loss from the Gaussian mechanism that is used in DG-SGD, and is a common analysis tool for DP-SGD literature. To compute the final (ϵ, δ)-differential privacy parameters from iterative runs of DP-SGD, one must first compute the subsampled Renyi Divergence, then compose privacy under RDP, and then convert the RDP guarantee into DP.

Step 1: Subsampled renyi divergence. Given sampling rate q and noise multiplier ψ, one can obtain RDP privacy parameters as a function of $\alpha \geqslant 1$ for one run of DP-SGD [23]. We denote this function by $\text{RDP}_{T=1}(\cdot)$, which will depend on q and ψ.

Step 2: Composition of RDP. When DP-SGD is run iteratively, we can compose the Renyi privacy parameter across all runs using the following proposition.

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

Hence, we can compute RDP privacy parameters for T iterations of DP-SGD as $\text{RDP-ACCOUNT}(T, q, \psi) := T \cdot \text{RDP}_{T=1}(\cdot)$.

Step 3: Conversion to (ϵ, δ)-DP. After obtaining an expression for the overall RDP privacy parameter values, any (α, ϵ)-RDP guarantee can be converted into (ϵ, δ)-DP.

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

Since the ϵ privacy parameter of RDP is also a function of α, this last step involves optimizing for the α that achieves smallest privacy parameter in Proposition 2.

3. Algorithmic framework

The overview of our algorithmic framework DP-auto-GAN is shown in Fig. 1, and the full details are given in Algorithm 2.

The algorithm takes in m raw data points, and *preprocesses* these points into m vectors $x_1, \ldots, x_m \in$

$\mathbb{R}^n$ to be read by DP-auto-GAN, where usually n is very large. For example, categorical data may be preprocessed using one-hot encoding, or text may be converted into high-dimensional vectors. Similarly, the output of DP-auto-GAN can be *post-processed* from $\mathbb{R}^n$ back to the data's original form. We assume that this pre- and post-processing can done based on public knowledge, such as possible categories for qualitative features and reasonable bounds on quantitative features, and therefore do not incur a privacy cost.

Within the DP-auto-GAN, there are two main components: the *autoencoder* and the GAN. The autoencoder serves to reduce the dimension of the data to $d \ll n$ before it is fed into the GAN. The GAN consists of a *generator* that takes in noise z sampled from a distribution Z and produces $G_w(z) \in \mathbb{R}^d$, and a *discriminator* $D_y(\cdot) : \mathbb{R}^n \to \{0, 1\}$. Because of the autoencoder, the generator only needs to synthesize data based on the latent distribution in $\mathbb{R}^d$, which is much easier than synthesizing in $\mathbb{R}^n$. Both components of our architecture, as well as our algorithm's overall privacy guarantee, are described in the remainder of this section.

3.1. Autoencoder framework and training

An autoencoder consists of an encoder $En_\phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^d$ and a decoder $De_\theta(\cdot) : \mathbb{R}^d \to \mathbb{R}^n$ parametrized by weights ϕ, θ respectively. The architecture of the autoencoder assumes that high-dimensional data $x_i \in \mathbb{R}^n$ can be represented compactly in a low-dimensional *latent space* $\mathbb{R}^d$. The encoder En_ϕ is trained to find such low-dimensional representations, and the decoder De_θ maps $En_\phi(x_i)$ in the latent space back to x_i. A natural measure of the information preserved in this process is the error between the decoder's image and the original x_i. A good autoencoder should minimize the distance $\mathrm{dist}(De_\theta(En_\phi(x_i)), x_i)$ for each point x_i for an appropriate distance function dist. Our autoencoder uses binary cross entropy loss: $\mathrm{dist}(x, y) = -\sum_{j=1}^n y_{(j)} \log(x_{(j)}) - \sum_{j=1}^n (1 - y_{(j)}) \log(1 - x_{(j)})$, where $x_{(j)}$ is the jth coordinate of the data $x \in [0, 1]^n$ after our preprocessing.

This motivates a definition of a (true) loss function $\mathbb{E}_{x \sim Z_X}[\mathrm{dist}(De_\theta(En_\phi(x)), x)]$ when data are drawn independently from an underlying distribution Z_X. The corresponding empirical loss function when we have an access to sample $\{x_i\}_{i=1}^m$ is

$$L_{\text{auto}}(\phi, \theta) := \sum_{i=1}^m \mathrm{dist}(De_\theta(En_\phi(x_i)), x_i). \quad (2)$$

Finding a good autoencoder requires optimizing ϕ and θ to yield small empirical loss in Eq. (2).

Algorithm 2: DPautoGAN (full procedure)

architecture input: Sensitive dataset $D \in \mathcal{X}^m$ where $\mathcal{X}$ is the (raw) data universe, preprocessed data dimension n, latent space dimension d, preprocessing function $Pre : \mathcal{X} \to \mathbb{R}^n$, post-processing function $Post : \mathbb{R}^n \to \mathcal{X}$, encoder architecture $En_\phi : \mathbb{R}^n \to \mathbb{R}^d$ parameterized by ϕ, decoder architecture $De_\theta : \mathbb{R}^d \to \mathbb{R}^n$ parameterized by θ, generator's noise distribution Z on sample space $\Omega(Z)$, generator architecture $G_w : \Omega(Z) \to \mathbb{R}^d$ parameterized by w, discriminator architecture $D_y : \mathbb{R}^n \to \{0, 1\}$ parameterized by y.
autoencoder training parameters: Learning rate η_1, number of iteration rounds (or optimization steps) T_1, loss function L_{auto}, optimization method $\text{OPTIM}_{\text{auto}}$ batch sampling rate q_1 (for batch expectation size $b_1 = q_1 m$), clipping norm C_1, noise multiplier ψ_1, microbatch size r_1
generator training parameters: Learning rate η_2, batch size b_2, loss function L_G, optimization method OPTIM_G, number of generator iteration rounds (or optimization steps) T_2
discriminator training parameters: Learning rate η_3, number of discriminator iterations per generator step t_D, loss function L_D, optimization method OPTIM_D, batch sampling rate q_3 (for batch expectation size $b_3 = q_3 m$), clipping norm C_3, noise multiplier ψ_3, microbatch size r_3
privacy parameter $\delta > 0$
procedure DPautoGAN
  $X \leftarrow Pre(D)$
  Initialize ϕ, θ, w, y for $En_\phi, De_\theta, G_w, D_y$
  ▷ *Phase 1: autoencoder training*
for $t = 1 \ldots T_1$ **do**
    $\text{DPTRAIN}_{\text{AUTO}}(X, En, De,$ autoencoder training parameters$)$
  ▷ *Phase 2: GAN training*
for $t = 1 \ldots T_2$ **do**
    for $j = 1 \ldots t_D$ **do**
    ▷ *(privately) train D_y for t_D iterations*
      $\text{DPTRAIN}_{\text{DISCRIMINATOR}}(X, Z, G, De, D,$ discriminator training parameters$)$
    $\text{TRAIN}_{\text{GENERATOR}}(Z, G, De, D,$ generator training parameters$)$
  ▷ *Privacy accounting*
$\text{RDP}_{\text{auto}}(\cdot) \leftarrow \text{RDP-ACCOUNT}(T_1, q_1, \psi_1, r_1)$
$\text{RDP}_D(\cdot) \leftarrow \text{RDP-ACCOUNT}(T_2 \cdot t_D, q_3, \psi_3, r_3)$
$\epsilon \leftarrow \text{GET-EPS}(\text{RDP}_{\text{auto}}(\cdot) + \text{RDP}_D(\cdot))$
return model (G_w, De_θ), privacy (ϵ, δ)

We minimize Eq. (2) privately using DP-SGD (Section 2.1). Our approach follows previous work on private training of autoencoders [10,11,16] by adding noise to both the encoder and decoder. In our DP-auto-GAN framework, the autoencoder is trained first until completion, and is then fixed while training the GAN. As noted earlier, the decoder is trained privately by clipping gradient norm and injecting Gaussian noise in order to obtain the gradient of decoder g_θ, while the gradient of encoder g_ϕ can be used directly as encoder can be trained non-privately.

3.2. GAN framework and training

A GAN consists of a generator G_w and a discriminator $D_y : \mathbb{R}^n \to \{0, 1\}$, parameterized respectively

Algorithm 3: DPTRAIN$_{\text{AUTO}}$(X, En_ϕ, De_θ, training parameters)

training parameter input: Learning rate η_1, number of iteration rounds (or optimization steps) T_1, loss function L_{auto}, optimization method OPTIM$_{\text{auto}}$ batch sampling rate q_1 (for the batch expectation size $b_1 = q_1 m$), clipping norm C_1, noise multiplier ψ_1, microbatch size r_1
goal: train one step of autoencoder (En_ϕ, De_θ)
procedure DPTRAIN$_{\text{AUTO}}$
    $\mathcal{B} \leftarrow$ SAMPLEBATCH(X, q_1)
    Partition $\mathbb{R}$ into $B_1, \dots, B_k$ each of size r (ignoring the dividend)
    $\hat{k} \leftarrow \frac{q_1 m}{r}$ ▷ an estimate of k
    for $j = 1 \dots k$ **do**
        ▷ *Both g_ϕ^j, g_θ^j can be computed in one backpropagation*
        $g_\phi^j, g_\theta^j \leftarrow \nabla_\phi(L_{\text{auto}}(De_\theta(En_\phi(B_j)), B_j)),$ $\nabla_\theta(L_{\text{auto}}(De_\theta(En_\phi(B_j)), B_j)$
    $g_\phi \leftarrow \frac{1}{\hat{k}} \sum_{j=1}^{k} g_\phi^j$
    $g_\theta \leftarrow \frac{1}{\hat{k}} \left(\left(\sum_{j=1}^{k} \text{CLIP}(g_\phi^j, C_1) \right) + \mathcal{N}(0, C_1^2 \psi_1^2 I) \right)$
    $(\phi, \theta) \leftarrow$ OPTIM$_{\text{auto}}(\phi, \theta, g_\phi, g_\theta, \eta_1)$

by weights w and y. The aim of the generator G_w is to synthesize (fake) data similar to the real dataset, while the discriminator aims to determine whether an input x_i is from the generator's synthesized data (and assign label $D_y(x_i) = 0$) or is real data (and assign label $D_y(x_i) = 1$). The generator is seeded with a random noise $z \sim Z$ that contains no information about the real dataset, such as a multivariate Gaussian vector, and aims to generate a distribution $G_w(z)$ that is hard for D_y to distinguish from the real data. Hence, the generator wants to minimize the probability that D_y makes a correct guess, $\mathbb{E}_{z\sim Z}[1 - D_y(G_w(z))]$. The discriminator wants to maximize its probability of a correct guess, which is $\mathbb{E}_{z\sim Z}[1 - D_y(G_w(z))]$ when the datum is fake and $\mathbb{E}_{x\sim Z_X}[D_y(x)]$ when it is real.

We extend the binary output of D_y to a continuous range $[0, 1]$, with the value indicating the confidence that a sample is real. We use the zero-sum objective for the discriminator and generator [34], which is motivated by the Wasserstein distance of two distributions. Although the proposed Wasserstein objective cannot be computed exactly, it can be approximated by optimizing:

$$\min_y \max_w O(y, w) := \mathbb{E}_{x\sim Z_X}[D_y(x)] - \mathbb{E}_{z\sim Z}[D_y(G_w(z))]. \tag{3}$$

We optimize Eq. (3) privately using the DP-SGD framework described in Section 2.1. We differ from prior work on DP-GANs in that our generator $G_w(\cdot)$ outputs data $G_w(z)$ in the latent space $\mathbb{R}^d$, which needs to be decoded by the fixed (pre-trained) De_θ to $De_\theta(G_w(z))$ before being fed into the discriminator $D_y(z)$. The gradient $\nabla_w G_w$ is obtained by backpropagation through this additional component $De_\theta(\cdot)$.

As suggested by [24], the discriminator trained for several iterations per one iteration of generator training. While the discriminator is being trained, the generator is fixed, and vice-versa. The discriminator and generator training are described in Algorithms 4 (DPTRAIN$_{\text{DISCRIMINATOR}}$) and 5 (TRAIN$_{\text{GENERATOR}}$) respectively. Since the discriminator receives real data samples as input for training, the training is made differentially private by clipping the norm of the gradient updates, and adding Gaussian noise to the gradient g. The generator does not use any real data in training (or any functions of the real data that were computed without differential privacy), and hence it can be trained without any need to clip the gradient norm or to inject noise into the gradient.

Algorithm 4: DPTRAIN$_{\text{DISCRIMINATOR}}$(X, Z, G_w, De_θ, D_y, training parameters)

training parameter input: Learning rate η_3, number of discriminator iterations per generator step t_D, loss function L_D, optimization method OPTIM$_D$, batch sampling rate q_3 (for the batch expectation size $b_3 = q_3 m$), clipping norm C_3, noise multiplier ψ_3, microbatch size r_3
goal: train one step of discriminator D_y
procedure DPTRAIN$_{\text{DISCRIMINATOR}}$
    $\mathcal{B} \leftarrow$ SAMPLEBATCH(X, q_3)
    Partition $\mathbb{R}$ into $B_1, \dots, B_k$ each of size r (ignoring the dividend)
    $\hat{k} \leftarrow \frac{q_1 m}{r}$ ▷ an estimate of k
    for $j = 1 \dots k$ **do**
        $\{z_i\}_{i=1}^r \sim Z^r$
        $B' \leftarrow \{De(G_w(z_i))\}_{i=1}^r$
        $g^j \leftarrow \nabla_y(L_D(B_j, B', D_y))$
        ▷ In the case of WGAN,
        $$L_D(B_j, B', D_y) := \frac{1}{r} \sum_{b \in B_j} D_y(b) - \frac{1}{r} \sum_{b' \in B'} D_y(b')$$
    $g \leftarrow \frac{1}{\hat{k}} \left(\left(\sum_{j=1}^{k} \text{CLIP}(g^j, C_3) \right) + \mathcal{N}(0, C_3^2 \psi_3^2 I) \right)$
    $y \leftarrow$ OPTIM$_D(y, g, \eta_3)$

After this two-phase training (of the autoencoder and GAN), the noise distribution Z, trained generator $G_w(\cdot)$, and trained decoder $De_\theta(\cdot)$ are released to the public. The public can sample $z \sim Z$ to obtain a synthesized datapoint $De_\theta(G_y(z))$ repeatedly to obtain a synthetic dataset of any desired size.

3.3. *Privacy accounting*

We use Renyi Differential Privacy (RDP) of [23], to account for privacy in each phase of training as in prior

Algorithm 5: $\text{TRAIN}_{\text{GENERATOR}}(Z, G_w, De_\theta, D_y,$ generator training parameters)

1: **training parameter input**: Learning rate η_2, batch size b_2, loss function L_G, optimization method OPTIM_G, number of generator iteration rounds (or optimization steps) T_2
2: **goal**: train one step of generator G_w
3: **procedure** $\text{TRAIN}_{\text{GENERATOR}}$
4: $\quad \{z_i\}_{i=1}^{b_2} \sim Z^{b_2}$
5: $\quad B' \leftarrow \{De(G_w(z_i))\}_{i=1}^{b_2}$
6: $\quad g \leftarrow \nabla_w(L_G(B', D_y))$
$\quad\quad \triangleright$ In the case of WGAN,
$$L_G(B', D_y) := -\frac{1}{b_2}\sum_{b' \in B'} D_y(b')$$
7: $\quad w \leftarrow \text{OPTIM}_G(w, g, \eta_2)$

works. Our autoencoder and GAN are trained privately by clipping gradients and adding noise to the encoder, decoder, and discriminator. Since the generator only accesses data through the discriminator's (privatized) output and De_θ is first trained privately and then fixed during GAN training, the trained parameters of generator are also private by post-processing guarantees of differential privacy. Privacy accounting is therefore required for only two parts that access real data X: training of the autoencoder and of the discriminator. In each training procedure, we apply the RDP accountant described in Section 2.2, to analyze privacy of the DP-SGD training algorithm.

The RDP accountant is a function $r : [1, \infty) \to \mathbb{R}_+$ and guarantees (ϵ, δ)-DP for any given $\delta > 0$ with $\epsilon = \min_{\alpha>1} r(\alpha) + \frac{\log 1/\delta}{\alpha-1}$ ([23]; also used in Tensorflow Privacy [49]). Hence, at the end of two-phase training, we have two RDP accountants r_1, r_2. We compose two RDP accountants *before* converting the combined accountant into (ϵ, δ)-DP. Note that another method used in DP-SYN [10] first converts r_i to (ϵ_i, δ_i)-DP and then combines them into $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP by basic composition [5]. For completeness, we show that composing RDP accountants first always results in a better privacy analysis.

Lemma 1. Let $\mathcal{M}_1, \mathcal{M}_2$ be any mechanisms and $r_1, r_2 : [1, \infty) \to \mathbb{R}_+ \cup \{\infty\}$ be functions such that $\mathcal{M}_1, \mathcal{M}_2$ are $(\alpha, r_1(\alpha))$- and $(\alpha, r_2(\alpha))$-RDP, respectively. Let $\delta \in (0, 1]$ and let

$$\epsilon_1 = \min_{\alpha>1} r_1(\alpha) + \frac{\log(2/\delta)}{\alpha-1},$$

$$\epsilon_2 = \min_{\alpha>1} r_2(\alpha) + \frac{\log(2/\delta)}{\alpha-1},$$

and

$$\epsilon = \min_{\alpha>1} r_1(\alpha) + r_2(\alpha) + \frac{\log(1/\delta)}{\alpha-1}.$$

Then $\mathcal{M}_1$ is $(\epsilon_1, \delta/2)$-DP, $\mathcal{M}_2$ is $(\epsilon_2, \delta/2)$-DP, and the composition $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2)$ is (ϵ, δ)-DP. If ϵ_1 and ϵ_2 are finite, then $\epsilon < \epsilon_1 + \epsilon_2$.

Proof. Let

$$\alpha_1^* \in \arg\min_{\alpha>1} r_1(\alpha) + \frac{\log(2/\delta)}{\alpha-1}$$

and

$$\alpha_2^* \in \min_{\alpha>1} r_2(\alpha) + \frac{\log(2/\delta)}{\alpha-1}$$

and let $\alpha = \min\{\alpha_1^*, \alpha_2^*\}$. Then, we have

$$\begin{aligned}\epsilon &\leqslant r_1(\alpha) + r_2(\alpha) + \frac{\log(1/\delta)}{\alpha-1}\\ &\leqslant r_1(\alpha_1^*) + r_2(\alpha_2^*) + \frac{\log(1/\delta)}{\alpha-1}\\ &= \epsilon_1 + \epsilon_2 + \frac{\log(1/\delta)}{\alpha-1} - \frac{\log(2/\delta)}{\alpha_1^*-1}\\ &\quad - \frac{\log(2/\delta)}{\alpha_2^*-1} < \epsilon_1 + \epsilon_2\end{aligned}$$

where the two inequalities use the definitions of ϵ_1, ϵ_2, ϵ, and the second inequality uses the fact that r_i is an increasing function of α [50]. □

For most settings of training parameters, we found that ϵ by RDP composition in Lemma 1 is $\approx 30\%$ smaller than that of the standard composition (see Fig. 2 for this privacy saving in our DP-auto-GAN $\epsilon = 0.51$ ADULT setting). The observation can be support by theoretical analysis as follows. It is observed in [51] that $r_i(\alpha)$ appears linear until a phase transition at some α, and is close to linear again. In our parameter settings, the optimal order to achieve smallest ϵ is before the phase transition, and thus $r_i(\alpha)$ "practically" behaves linear as the privacy analysis never uses $r_i(\alpha)$ at α beyond the phase transition. This is illustrated in Fig. 3 by an example of our DP-auto-GAN $\epsilon = 0.51$ ADULT setting.

Assuming linear $r_i(\alpha) = c_i\alpha$, we can compute the analytical solutions:

$$\epsilon_1 = c_1 + 2\sqrt{c_1 \log 2/\delta},$$

$$\epsilon_2 = c_2 + 2\sqrt{c_2 \log 2/\delta},$$

$$\epsilon = c_1 + c_2 + 2\sqrt{(c_1 + c_2)\log 1/\delta}$$

In practice, δ is small compared to c_i's and the term $\sqrt{c_i \log 1/\delta}$ dominates. Hence, $\epsilon^2 \approx \epsilon_1^2 + \epsilon_2^2$, and for many settings where we set ϵ_1 close to ϵ_2 (such as in our setting or DP-SYN [10]), this implies $\epsilon \approx 0.707(\epsilon_1 + \epsilon_2)$, an approximately 30% reduction of privacy cost.

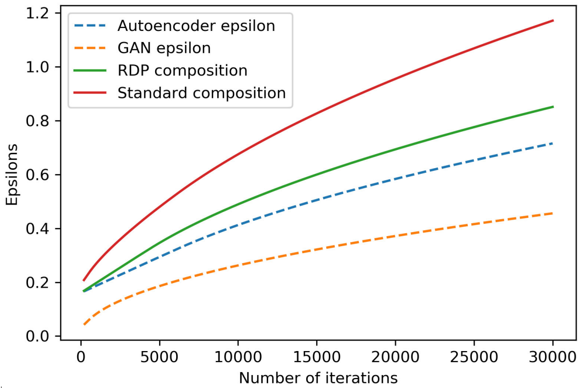


Fig. 2. Privacy ϵ for different training phases of the algorithm in $\epsilon = 0.51$ DP-auto-GAN parameter setting for ADULT data: the sampling rate q and noise multiplier ψ for autoencoder and GAN are $q = \frac{64}{32561}$, $\psi = 2.5$ and $q = \frac{128}{32561}$, $\psi = 7.5$, respectively. We target $\delta = 10^{-5}$ overall and $\delta = \frac{1}{2} \cdot 10^{-5}$ for each training phase.

4. Evaluation metrics for synthetic data

In this section, we review the evaluation schemes for measuring quality of synthetic data in existing literature. Various evaluation metrics have been considered in the literature to quantify the quality of synthetic data [46]. Broadly, evaluation metrics can be divided into two major categories: supervised and unsupervised. Supervised evaluation metrics are used when clear distinctions exist between features and labels in the dataset, e.g., for healthcare applications, whether a person has a disease or not could be a natural label. Unsupervised evaluation metrics are used when no feature of the data can be decisively termed as a label. For example, a data analyst who wants to learn a pattern from synthetic data may not know what specific prediction tasks to perform, but rather wants to explore the data using an unsupervised algorithm such as Principle Component Analysis (PCA). Unsupervised metrics can then be divided into three broad types: prediction-based, distributional-distance-based, and qualitative (or visualization-based). We describe supervised evaluation metrics and all three types of unsupervised evaluation metrics below. Metrics in previous work and our proposed metrics in this paper are summarized in Table 2.

Various evaluation metrics have been considered in the literature to evaluate the quality of the synthetic data (see Charest [46] for a survey). The metrics can be broadly categorized into two groups: *supervised* and *unsupervised*. Supervised evaluation metrics are used when there are clear distinctions between features and labels of the dataset, e.g., for healthcare applications, a person's disease status is a natural label. In these settings, a predictive model is typically trained on the synthetic data, and its accuracy is measured with respect to the real (test) dataset. Unsupervised evaluation metrics are used when no feature of the data can be decisively termed as a label. Recently proposed metrics include *dimension-wise probability* for binary data [47], which compares the marginal distribution of real and synthetic data on each individual feature, and *dimension-wise prediction*, which measures how closely synthetic data captures relationships between features in the real data. This metric was proposed for binary data, and we extend it here to mixed-type data. Recently, NIST [44] used a 3-way marginal evaluation metric which used three random features of the real and synthetic datasets to compute the total variation distance as a statistical score.

Supervised evaluation metrics. The main aim of generating synthetic data in a supervised setting is to best understand the relationship between features and labels. A popular metric for such cases is to train a machine learning model on the synthetic data and report its accuracy on the real test data [15]. Zhang et al. [43] used inception scores on the image data with classification tasks. Inception scores were proposed in Salimans et al. [27] for images which measure quality as well as diversity of the generated samples. Another metric used in Jordon et al. [41] reports whether the accuracy ranking of different machine learning models trained on the real data is preserved when the same machine learning model is trained on the synthetic data. Although these metrics are used for classification in the literature, they can be easily generalized to the regression setting.

In the DP setting of synthetic data generation, supervised metrics also differ from unsupervised in that the label feature is sometimes treated as public (e.g. in DP-SYN [10]), whereas in unsupervised setting, all features are treated as private. We note it as this may create a slight difference in privacy accounting.

Unsupervised evaluation metrics, prediction-based. Rather than measuring accuracy by predicting one particular feature as in supervised-setting, one can predict *every* individual feature using the rest of features. The prediction score is therefore created for each single feature, creating a list of dimension- (or feature-) wise prediction scores. Good synthetic data should have similar dimension-wise prediction scores to that of the real data. Intuitively, similar dimension-wise prediction shows that synthetic data correctly captures inter-feature relationships in the real data.

One metric of this type is proposed by Choi et al. [47] for binary data. Although it was originally proposed

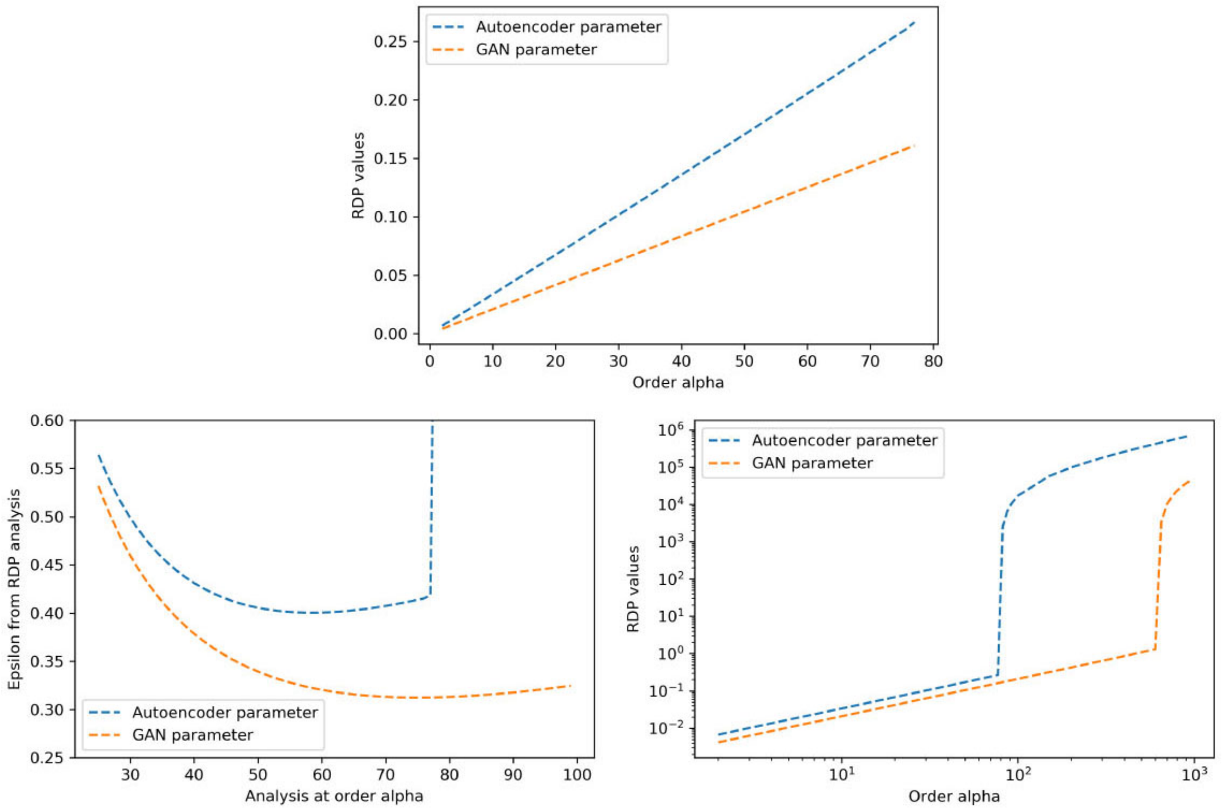


Fig. 3. RDP values over different order α for $\epsilon = 0.51$ DP-autho-GAN parameter setting for ADULT data: the sampling rate q, noise multiplier ψ, and T the number of training iterations for autoencoder and GAN are $q = \frac{64}{32561}$, $\psi = 2.5$, $T = 10000$ and $q = \frac{128}{32561}$, $\psi = 7.5$, $T = 15000$, respectively. The phase transitions (spikes of RDP value) for autoencoder and GAN appear at $\alpha = 79$ and $\alpha = 624$. The optimal order for smallest ϵ for autoencoder and GAN analysis targeting $\delta = \frac{1}{2}10^{-5}$ are 60 and 77, below the phase transitions.

for binary data, we extend this to mixed-type data by allowing varieties of predictive models appropriate for each data type present in the dataset. For each feature, we try predictive models on the real dataset in order of increasing complexity until a good accuracy score is achieved. For example, to predict a real-valued feature, we first used a linear classifier and then a neural network predictor. This ensures that a choice of predictive model is appropriate to the feature. Synthetic data is then evaluated by measuring the accuracy of the same predictive model (trained on the real data) on the synthetic data. Similarly high accuracy scores on synthetic data and real data indicates that the synthetic data closely approximates the real data.

Zhang et al. [43] provides an unsupervised Jensen-Shannon score metric which measures the Jensen-Shannon divergence between the output of a discriminating neural network on the real and synthetic datasets, and a Bernoulli random variable with 0.5 probability. This metric differs from dimension-wise prediction in that the predictive model (discriminator) is trained over the whole dataset at once, rather than dimension-wise, to obtain a score.

Diversity metric. Inception score is one common metric for evaluating the quality of data generated by GAN [52]. Both inception and Jensen-Shannon scores aim to capture both the accuracy and diversity of generated data through comparing the distributions of predictions by a fixed classifier on original and synthetic data. Inception score is similar to μ-smoothed KL divergence we propose in Section 5, but we apply it to discrete distribution and use a smoothing to avoid divergence being undefined. Our metric also differs from inception scores in that it is based on the distributions of synthetic and original data, and not on predictions on those datasets by any classifier. We observed that introducing a classifer can itself be a reason for lack of diversity, and concern that a predictive model in general can introduce bias and unfairness in other forms. For example, we found that in a categorical feature with one strong majority class, the classifier predicts only the majority to maximize a standard notion of "accuracy," hence making a synthetic data that ignore minority classes represent the original data perfectly well, as predictions on synthetic and original data are identical. Therefore, for diversity applications, we prefer distribution-based metric to a distribution-based metric.

Moreover, we aim our metric to be appropriate in differential privacy setting. A natural metric to penalize missing a minority class is KL divergence, as used in the definitions of inception and Jensen-Shannon scores. However, it is impossible for a private model to recognize if a minority exists if the class is really small, simply due to the definition of differential privacy (unless the algorithm assumes existence of all possible classes in the dataset, but this would greatly impact accuracy as the number of classes increase). Bagdasaryan et al. [53] observed a phenomena that differentially private training indeed impacts minority classes more than majority class, as we also observed in our work. Missing a minority class, therefore, is sometimes unavoidable with DP guarantees. Since missing any class makes KL divergence undefined, we added a smoothing term to KL divergence so that the penality of missing a minority class is finite, yet significant.

Unsupervised evaluation metrics, distributional-based. One way to evaluate the quality of synthetic data is computing a dimension-wise probability distribution, which was also proposed in Choi et al. [47] for binary data. This metric compares the marginal distribution of real and synthetic data on each individual feature. Below we survey other metrics in this class that can extend to mixed-type data.

3-Way Marginal: Recently, the NIST [44] challenge used a 3-way marginal evaluation metric in which three random features of the real and synthetic data R, S are used to compute the total variation distance as a statistical score. This process is repeated a few times and finally, average score is returned. In particular, values for each of the three features are partitioned in 100 disjoint bins as follows:

$$B^i_{R,k} = \left\lfloor \frac{(R^i_k - R_{k,\min}) * 100}{R_{k,\max} - R_{k,\min}} \right\rfloor$$

and

$$B^i_{S,k} = \left\lfloor \frac{(S^i_k - R_{k,\min}) * 100}{R_{k,\max} - R_{k,\min}} \right\rfloor,$$

where R^i_k, S^i_k is the value of i-th datapoint's k-th feature in datasets R and S, and $R_{k,\min}, R_{k,\max}$ are respectively the minimum and maximum value of the k-th feature in R. For example, if $k = 1, 2, 3$ are the selected features then i-th data points of R and S are put into bins identified by a 3-tuple, $(B^i_{R,1}, B^i_{R,2}, B^i_{R,3})$ and $(B^i_{S,1}, B^i_{S,2}, B^i_{S,3})$, respectively.

Let $\mathcal{B}_R, \mathcal{B}_S$ be the set of all 3-tuple bins in datasets R and S, and let $|B|$ denote number of datapoints in 3-tuple bin B, normalized by total number of data points. Then, the 3-way marginal metric reports the ℓ_1-norm of the bin-wise difference of $\mathcal{B}_R$ and $\mathcal{B}_S$ as follows:

$$\begin{aligned}
&\sum_{B_1 \in \mathcal{B}_R} \sum_{B_2 \in \mathcal{B}_S} \mathbb{I}_{\{B_1 \in \mathcal{B}_S\}} \mathbb{I}_{\{B_2 = B_1\}} ||B_1| - |B_2|| \\
&\quad + \sum_{B_1 \in \mathcal{B}_R} (1 - \mathbb{I}_{\{B_1 \in \mathcal{B}_S\}}) |B_1| \\
&\quad + \sum_{B_2 \in \mathcal{B}_S} (1 - \mathbb{I}_{\{B_2 \in \mathcal{B}_R\}}) |B_2|.
\end{aligned}$$

Both aforementioned metrics (dimension-wise probability from [47] and 3-way marginal from [44]) involve two steps. First, a projection (or a selection of features) of data is specified, and second some statistical distance or visualization of synthetic and real data in the projected space is computed. Dimension-wise probability for binary data corresponds to projecting data into each single dimension, and visualizing synthetic and real distributions in projected space by histograms (for binary data, the histogram can be specified by one single number: probability of the feature being 1). The 3-way marginal metric first selects a three-dimensional space specified by three features as a space into which data projected, discretizes the synthetic and real distributions on that space, then computes a total variation distance between discretized distributions. We can generalize these two steps process and conceptually design a new metric as follows.

Generalization of data projection: One can generalize selection of 3 features (3-way marginal) to any k features (k-way marginal). However, one can also select *k principle components* instead of k features. We distinguish these as k-way *feature* marginal (projection onto a space spanned by feature dimensions) and k-way *PCA* marginal (projection onto a space spanned by principle components of the original dataset). Intuitively, k-way PCA marginal best compresses the information of the real data into a small k-dimensional space, and hence is a better candidate for comparing projected distributions.

Generalization of distributional distance: Total variation distance can be misleading as it does not encode any information on the distance between the supports of two distributions. In general, one can define any metric of choice (optionally with discretization) on two projected distributions, such as Wasserstein distance which also depends on the distance between the supports of the two distributions.

Computing distributional distance: The distance between two distributions can also be computed without any data projections. Computing an exact statistical score on high-dimensional datasets is likely computationally hard. However, one can, for example, subsample uniformly at random points from two distributions to compute the score more efficiently, then average this distance over many iterations.

Unsupervised evaluation metrics, qualitative. As described above, dimension-wise probability is a specific application of comparing histograms under binary data. One can plot histograms of each feature (1-way feature marginal) for inspection. In practice, histogram visualization is particularly helpful when a feature is strongly skewed, sparse (majority zero), and/or hard to predict well by predictive models. An example of this occurred when predictive models do not have meaningful predictive accuracy on certain features of the ADULT dataset, making prediction-based metric inappropriate. Instead, inspection of histograms of those features on synthetic and real data (as in Fig. 12) indicate that synthetic data replicates those features well.

In addition, 2-way PCA marginal is a visual representation of data that explains as much variance as possible in a plane, providing a good trade-off between information and ease of visualization on two datasets. This visualization can be augmented with a distributional distance of choice over the two distributions on these two spaces to get a quantitative metric.

4.1. Background on evaluation metrics used in experiments

Here, we discuss in more technical details the evaluation metrics that we use in the experiments in Section 5 and in our paper to empirically measure the quality of the synthetic data. Some of these metrics have been used in the literature, while 2-way PCA is novel in this work. Another novel metric μ-smooth KL divergence is described in Section 5.

For the following two metrics, the dataset should be partitioned into a training set $R \in \mathbb{R}^{m_1 \times n}$ and testing set $T \in \mathbb{R}^{m_2 \times n}$, where $m = m_1 + m_2$ is the total number of samples the real data, and n is the number of features in the data. After training, the generative model creates a synthetic dataset $S \in \mathbb{R}^{m_3 \times n}$ for sufficiently large m_3.

Dimension-wise probability. When the feature is binary, we compares the proportion of 1's (which can be thought of as estimators of Bernoulli success probability) in each feature of the training set R and synthetic dataset S, i.e. the marginal distribution of each feature. For each feature, the closer the proportion of 1's in the original dataset is to that of synthetic dataset, the better.

Dimension-wise prediction. This metric evaluates whether synthetic data maintain relationships *between* features. For the k-th feature of training set R and synthetic dataset S, we choose $y_{R_k} \in \mathbb{R}^{m_1}$ and $y_{S_k} \in \mathbb{R}^{m_2}$ as labels of a classification or regression task based on the type of that feature, and the remaining features R_{-k} and S_{-k} are used for prediction. We train either a classification or regression model on R_{-k} and S_{-k}, and measure goodness of fit based on the model's accuracy by testing on T. That is, we "train on synthetic, test on original" to evaluate the quality of synthetic data. The closer of accuracy scores obtained from original and synthetic data, the better.

Model accuracy can be reported using AUROC, F_1, or R^2 scores, as appropriate. We describe the model's accuracy as follows:

1. Area under the ROC curve (AUROC) score and F_1 score for classification: The F_1 score of a classifier is defined as $F_1 := \frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$, where precision is ratio of true positives to true and false positives, and recall is ratio of true positives to total true positives (i.e., true positives plus false negatives). F_1 score on multi-class features are averaged using micro-averaging. AUROC score is a graphical measure capturing the area under ROC (receiver operating characteristic) curve, and is only intended for binary data. Both metrics take values in interval [0, 1] with larger values implying good fit. The ROC curve are pairs of true and false positive rates obtained from setting different thresholds at the classifier's predicted probability. Note that when the classifier is trained on the data with one class and predicts always with probability 0 (or 1), ROC curve is a single pair, and AUROC is thus undefined.
2. R^2 score for regression: The R^2 score is defined as $1 - \dfrac{\sum(y_i - \widehat{y_i})^2}{\sum(y_i - \bar{y})^2}$, where y_i is the true label, $\widehat{y_i}$ is the predicted label, and $\bar{y}$ is the mean of the true labels. This is a popular metric used to measure goodness of fit as well as future prediction accuracy for regression.

1-Way feature marginal (histogram). We compute probability distribution of the feature of interest of both real and synthetic data. For continuous features, we partition the range into intervals. This can be extended to k-way feature marginals by considering joint distribution over k features and made into a quantitative measure by adding a distance measure between the histograms.

We also propose the following novel qualitative evaluation metric.

2-Way PCA marginal. This metric generalizes the 3-way marginal score used in NIST [44]. In particular, we compute principle components of the original data and evaluate a projection operator for first two principle components. Denote $P \in \mathbb{R}^{n\times 2}$ the projection matrix such that $\bar{R} = RP$ is the projection on first two principle components of R. After we fix P, we project synthetic data $\bar{S} = SP$ and scatterplot 2-D points in $\bar{R}$ and $\bar{S}$ for visual evaluation. That is, we train PCA from the original dataset, and use the same projection from this PCA on (possibly many) synthetic datasets.

5. Experiments

In this section, we empirically evaluate the performance of our DP-auto-GAN framework on the MIMIC-III [12] and ADULT [13] datasets, which have been used in prior works on differentially private synthetic data generation. We compare against these prior approaches using a variety of qualitative and quantitative evaluation metrics, including some from prior work and some novel metrics we introduce. We target $\delta = 10^{-5}$ in all settings, and all ϵ values reported that are rounded are rounded up to guarantee the validity of privacy guarantee. All experimental details and additional experimental results can be found in Appendices A and B, and our code is available at https://github.com/DPautoGAN/DPautoGAN.

5.1. Binary data

MIMIC-III [12] is a binary dataset consisting of medical records of 46,520 intensive care unit (ICU) patients over 11 years old with 1071 features. For the experiments, the data was partitioned in into train, validation, and test data sets of sizes 60%, 20%, 20%, respectively.

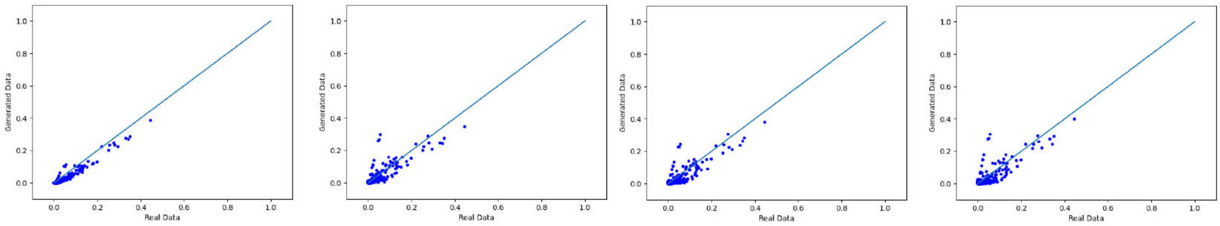
Even though DP-auto-GAN can handle mixed-type data, we evaluate it first on MIMIC-III since this dataset has been used in similar non-private [47] and private [15] GAN frameworks. We apply the same evaluation metrics used in these papers, namely dimension-wise probability and dimension-wise prediction. Prediction is defined by AUROC score of a logistic regression classifier.

Dimension-wise probability. Figure 4 shows the dimension-wise probability of DP-auto-GAN for different ϵ. Each point in the figure corresponds to a feature in the dataset, and the x and y coordinates respectively show the proportion of 1s in the real and synthetic datasets. Points closer to the $y = x$ line correspond to better performance, because this indicates the distribution is similar in the real and synthetic datasets. As shown in Fig. 4, the proportion of 1's in the marginal distribution for is similar on the real and synthetic datasets in the non-private ($\epsilon = \infty$) and private settings. The marginal distributions of the privately generated data from DP-auto-GAN remain a close approximation of the real dataset, even for small values of ϵ, because nearly all points fall close to the line $y = x$. We note that our results are significantly stronger than the ones obtained in [15] with $\epsilon \in [96.5, 231]$ because we obtain dramatically better performance with ϵ values that are two orders of magnitude smaller. For visual performance comparison, see Fig. 4 of [15].

Dimension-wise prediction. Figure 5 shows dimension-wise prediction using DP-auto-GAN for different values of ϵ. Each point in the figure corresponds to a feature in the dataset, and the x and y coordinates respectively show the AUROC score of a logistic regression classifier trained on the real and synthetic datasets, and points closer to the $y = x$ line still correspond to better performance. As shown in the figure, for $\epsilon = \infty$, many points are concentrated along the lower side of line $y = x$, which indicates that the AUROC score of the real dataset is only marginally higher than that of the synthetic dataset. When privacy is added, there is a gradual shift downwards relative to the line $y = x$, with larger variance in the plotted points, indicating that AUROC scores of real and synthetic data show more difference when privacy is introduced. Surprisingly, there is little degradation in performance for smaller ϵ values, including $\epsilon = 0.81$. For sparse features with few 1's in the data, the generative model will output all 0's for that feature, making AUROC ill-defined. We follow [15] by excluding those features from dimension-wise prediction plots.

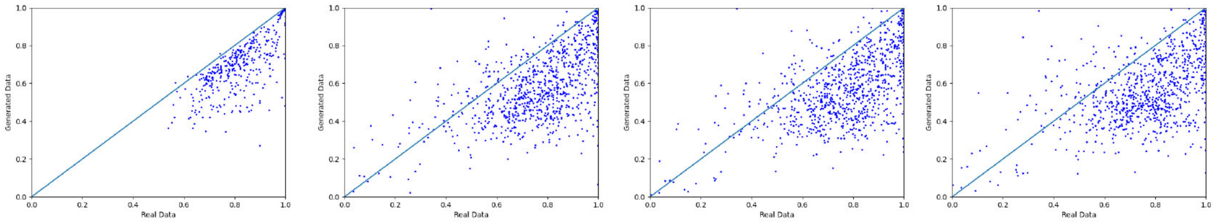
Our results for DP-auto-GAN under this metric are also significantly stronger than the ones obtained in [15] with much larger ϵ values of $\epsilon \in [96.5,231]$; for visual performance comparison, see Fig. 5 of [15]. Our probability and prediction plots of DP-auto-GAN are either comparable to or better than [15], with our prediction plots detecting many more sparse features. The performance of DP-auto-GAN degrades only slightly as ϵ decreases and is achieved at much smaller ϵ values, giving a roughly 100x improvement in privacy compared to [15].

Dimension-wise prediction for sparse features. MIMIC-III dataset contains several sparse features, i.e.



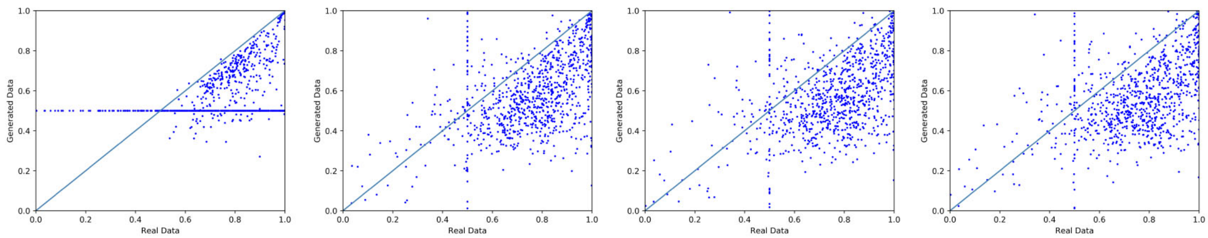
(a) $\epsilon = \infty$ (b) $\epsilon = 2.70$ (c) $\epsilon = 1.33$ (d) $\epsilon = 0.81$

Fig. 4. Dimension-wise probability scatterplots for different values of ϵ. Each point represents one of the 1071 features in the MIMIC-III dataset. The x and y coordinates of each point are the proportion of 1s in real and synthetic datasets of a feature, respectively. The line $y = x$, which represents ideal performance, is shown in each plot. Note that even for small ϵ values, performance is not degraded much relative to the non-private method. Compare with Fig. 4 in [15], which provides worse performance for $\epsilon \in [96, 231]$.



(a) $\epsilon = \infty$ (b) $\epsilon = 2.70$ (c) $\epsilon = 1.33$ (d) $\epsilon = 0.81$

Fig. 5. Dimension-wise prediction scatterplots for different values of ϵ. Each point represents one of the 1071 features in the MIMIC-III dataset. The x and y coordinates of each point represent the AUROC score of a logistic regression classifier trained on real and synthetic datasets, respectively. The line $y = x$ corresponds to the ideal performance. Again we note that even for small ϵ values, performance is not degraded much relative to the non-private method. Compare with Fig. 5 in [15], which provides worse performance for $\epsilon \in [96, 231]$.

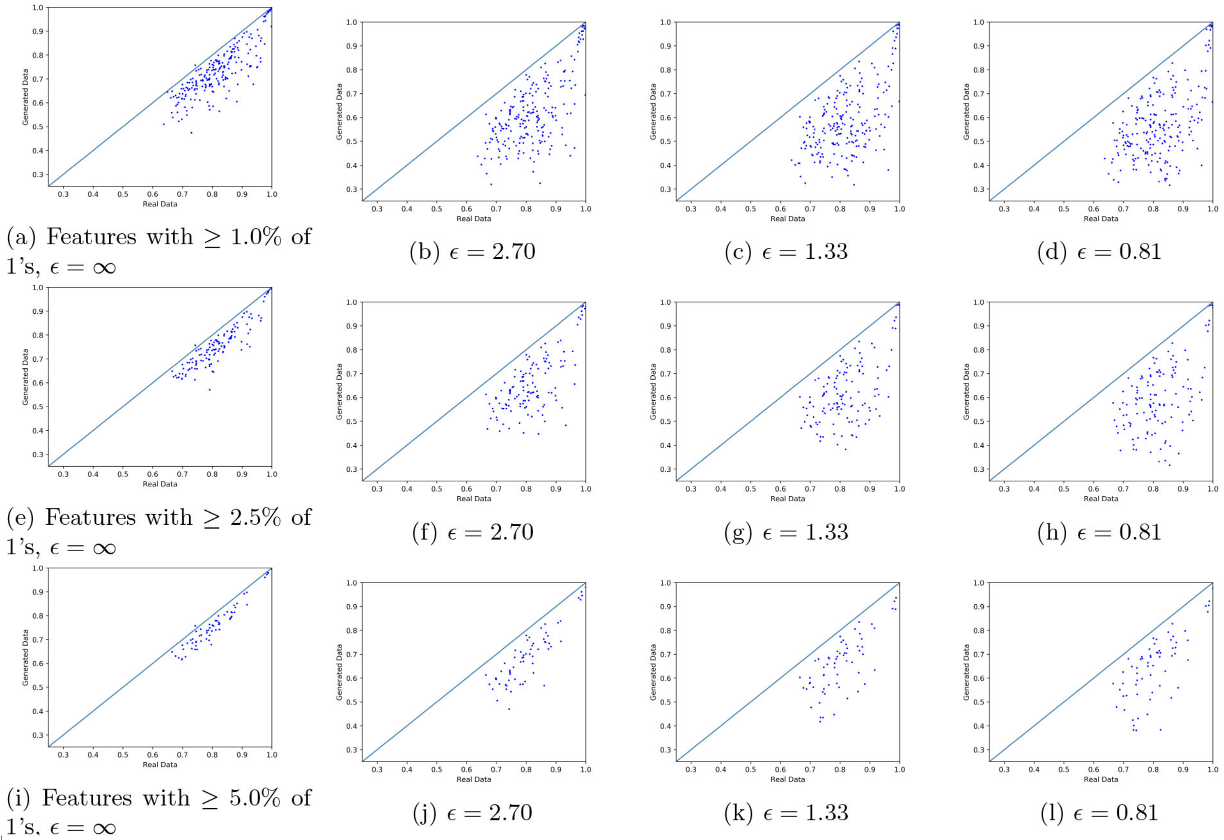


(a) $\epsilon = \infty$ (b) $\epsilon = 2.70$ (c) $\epsilon = 1.33$ (d) $\epsilon = 0.81$

Fig. 6. Dimension-wise prediction scatterplots of 925 of 1071 features which AUROC prediction scores are defined on the original dataset. AUROC is not defined when the test set of the original data has only one class on that feature. Prediction scores are by a logistic regression classifier trained on the original binary dataset MIMIC-III and on synthetic datasets generated by DP-auto-GAN at different privacy parameters ϵ.

features with small number of 1's. In fact, we found that 146 features do not have any 1's, and 706 more features have proportion of 1's less than 1% in the original dataset. The presence of sparse features is a challenge for prediction-based evaluation since the classifier accuracy is unstable on sparse features. Moreover, the prediction score is unmeaningful when the train or test dataset has only one class present, such as AUROC being undefined when one class is presented in the test data. Even when AUROC is defined, the sparsity of 1's in dataset makes a classifier unable to learn anything and give a score of 0.5, or perform worse than a random classifer and give a score below 0.5, which is arguably unmeaningful.

In our experiment, after an 80%/20% split into the training and testing datasets, AUROC scores are not defined on those 146 features with no 1's in the original dataset. Of the rest 925 features, 35 and 69 of those have AUROC prediction scores exactly at and below 0.5, respectively, on the original dataset. All 104 of



(a) Features with $\geq 1.0\%$ of 1's, $\epsilon = \infty$ (b) $\epsilon = 2.70$ (c) $\epsilon = 1.33$ (d) $\epsilon = 0.81$

(e) Features with $\geq 2.5\%$ of 1's, $\epsilon = \infty$ (f) $\epsilon = 2.70$ (g) $\epsilon = 1.33$ (h) $\epsilon = 0.81$

(i) Features with $\geq 5.0\%$ of 1's, $\epsilon = \infty$ (j) $\epsilon = 2.70$ (k) $\epsilon = 1.33$ (l) $\epsilon = 0.81$

Fig. 7. Dimension-wise prediction scatterplots of 219 of 1071 features with at least 1% of 1's, 127 features with at least 2.5% of 1's, and 64 features with at least 5% of 1's in the original dataset. Prediction scores are by a logistic regression classifier trained on the original binary dataset MIMIC-III and on synthetic datasets generated by DP-auto-GAN at different privacy parameters ϵ.

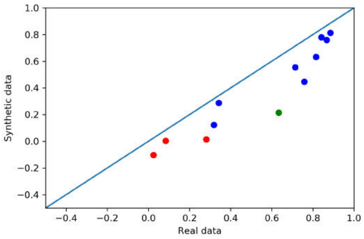
those with unmeaningful AUROC have less than 0.2% proportion of 1's in the original dataset. Figure 6 shows dimension-wise prediction scatterplots of full 925 of 1071 features where AUROC is defined.

When $\epsilon = \infty$, DP-auto-GAN generates all 0's in many of the sparse features in the synthetic dataset, including all 146 features with no 1's in the original datasets. Those features obtain scores of 0.5, giving the horizontal line $y = 0.5$ in Fig. 6a. When noise is injected, DP-auto-GAN generates few but enough of 1's in all features that AUROC is not 0.5, and thus the horizontal line is no longer present in Fig. 6b–d. The vertical line $x = 0.5$ represents 69 features that the classifier is unable to learn in the orignal dataset due to the sparsity and learn from random noise injected in the synthetic data generator.

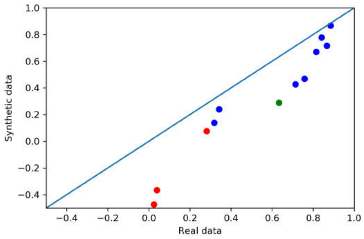
Xie et al. [15] presented dimension-wise prediction by deleting features which synthetic dataset contain no 1's. While this conveniently deletes points on the horizontal and vertical lines, the features being deleted are dependent on the synthetic dataset, which can obscure the presentation of its quality. For example, if the generator performs poorly on non-sparse features by outputting only one class, that feature is not on the plot rather than being presented as far away from the line $y = x$. Since we observe that prediction scores on sparse features are unstable, thus necessarily showing large variances in the context of differential privacy, we propose to delete features whose proportions of 1's in the original dataset are below a threshold. The set of deleted feature is therefore fixed and independent of synthetic data to be evaluated. In Fig. 7, we show dimension-wise prediction scatterplots of MIMIC-III dataset with the thresholds set at 1%, 2.5%, and 5%. We are able to more clearly see the performance of DP-auto-GAN than plotting all 925 features and better observe a slight change of performance as ϵ decreases across three ϵ values.

5.2. Mixed-type data

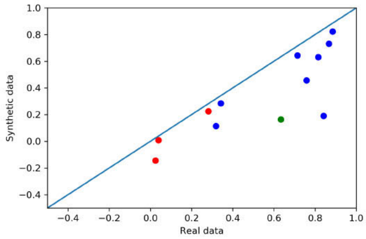
ADULT dataset [13] is an extract of the U.S. Census of 48K working adults, consisting of mixed-type data:



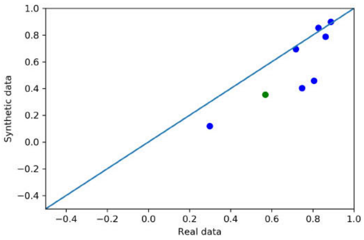
(a) DPautoGAN $\epsilon = 1.01$



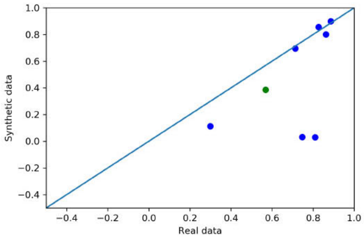
(b) DPautoGAN $\epsilon = 0.51$



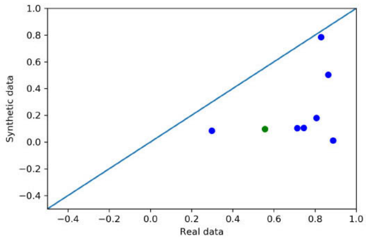
(c) DPautoGAN $\epsilon = 0.36$



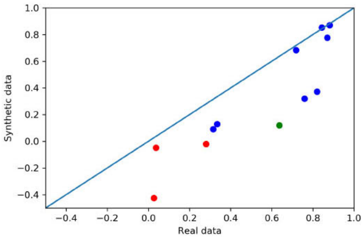
(d) DP-WGAN $\epsilon = 1.01$



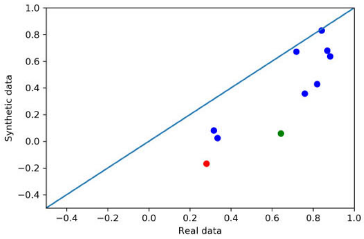
(e) DP-WGAN $\epsilon = 0.51$



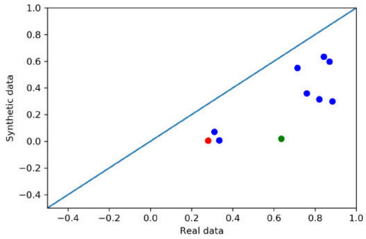
(f) DP-WGAN $\epsilon = 0.36$



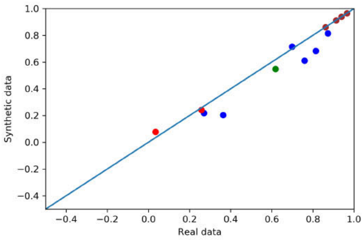
(g) DP-VAE $\epsilon = 1.01$



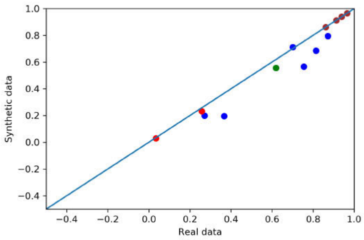
(h) DP-VAE $\epsilon = 0.51$



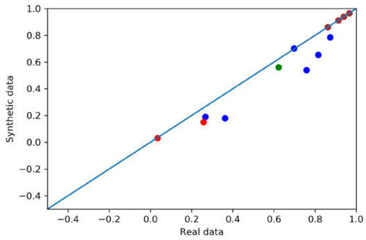
(i) DP-VAE $\epsilon = 0.36$



(j) DP-SYN $\epsilon = 1.40$



(k) DP-SYN $\epsilon = 0.80$



(l) DP-SYN $\epsilon = 0.50$

Fig. 8. Dimension-wise prediction scatterplot of all (applicable) features of ADULT dataset for different ϵ values and algorithms. The line $y = x$ represents ideal performance. Blue, green, and red points respectively correspond to unlabeled categorical, labeled binary, and continuous features. Brown points indicate the synthetic data exhibit no diversity (i.e., all data points have the same category). Note that DP-SYN has several features without diversity. Red points with R^2 scores close to zero in the original data have unstable (and unmeaningful) synthetic R^2 scores due to the sparse nature of those features in the original data, and some of these R^2 scores fall outside of the plotted range. The implementation of DP-WGAN in [14] did not allow continuous features, and the implementation of DP-SYN in [10] converted two continuous features to categorical; see Appendix B for more details.

nine categorical features (one of which is a binary label) and four continuous. This dataset has been used to evaluate DP-WGAN [14] and DP-SYN [10]. We compare DP-auto-GAN against these methods, as well as DP-VAE [16]. We target $\epsilon = 1.01, 0.51, 0.36$. For DP-SYN, we allow $\epsilon = 1.4$,0.8, 0.5 because their implementation uses standard privacy composition, which is looser than than RDP composition (Lemma 1). These larger ϵ values provide comparable privacy guarantees to the smaller ϵ values achieved by RDP composition, and allow for a fair comparison of architectures without modifying the implementation in [10]. For more details, see Appendix B.4.

Dimension-wise prediction. Figure 8 compares the performance of DP-auto-GAN with these three prior algorithms for the task of dimension-wise prediction. For categorical features (represented by blue points and a single green point), we use a random forest classifier for prediction as in [14], and we measure performance using F_1 score, which is more appropriate than AUROC for multi-class prediction. For continuous features (represented by red points), we used Lasso regression and report R^2 scores. The green point corresponds to the *salary feature* of the data, which is real-valued but treated as binary based on the condition $>$ \$50 k, which was similarly used as a binary label in [14]. We use brown points to indicate the categorical features for which the synthetic data exhibit no diversity, where all synthetic data points have the same category. We explore metrics for measuring diversity later in this section.

Note that in Fig. 8, there are not four red points in each plot (corresponding to the four continuous features of the dataset). While AUROC for the binary features is always supported on [0, 1], the R^2 score for real-valued features can be negative if the predictive model is poor, and these values for these missing points fell outside the range of Fig. 8. These features are explored later in Fig. 11, using 1-way marginals as a qualitative metric.

Each point in Fig. 8 corresponds to one feature, and the x and y coordinates respectively show the accuracy score on the real data and the synthetic data. Figure 8 shows that DP-auto-GAN achieves considerable performance for all ϵ values tested. As expected, its performance degrades as ϵ decreases, but not substantially. DP-WGAN [14] performs well at $\epsilon = 1.01$, but its performance degrades rapidly with smaller ϵ. This is consistent with [14], which uses higher $\epsilon = 3$, 7. DP-auto-GAN outperforms DP-VAE [16] across all ϵ values. DP-SYN [10] is able to capture relationships between features well even for small ϵ using this metric.

1-Way marginal and diversity divergence. While DP-SYN has good dimension-wise prediction, this does not capture *diversity*, a concern of bias known for DP-SGD ([53]). For features with a large majority class and many minority classes, the classifier often predicts the majority class with probability one. We found that for four features, DP-SYN generates data from only one class, whereas all other algorithms do not behave this way for any feature. Lack of diversity in synthetic data can raise fairness concerns, as societal decisions based on the private synthetic data will inevitably ignore minority groups.

We start by turning to 1-way marginal as a method of evaluation, which is able to detect such issues and give another perspective of synthetic data. Figure 9 shows histograms of synthetic data from the four algorithms on two categorical features: marital-status and race. Marital-status distributes more evenly across categories, and DP-VAE, DP-SYN and DP-auto-GAN are able to learn this distribution well. Race, on the other hand, has an 85.5% majority; DP-SYN only generated data from the majority class, whereas DP-auto-GAN and DP-VAE were able to detect the existence of minority classes. DP-WGAN suffered similar issues on the marital status feature.

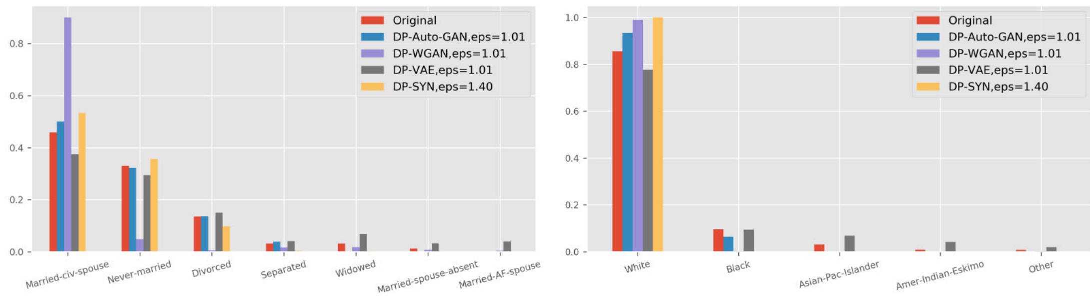
Figure 10 shows similar histograms for the native-country feature of the ADULT dataset, where the majority class constitutes $> 85\%$ of the population. Each of the 41 minority classes in native-country constitute less than 2% in the original dataset, with most of them weighing less than 0.2% of the population. DP-auto-GAN and DP-WGAN are able to capture some minority classes. DP-VAE was unable to accurately learn the majority structure and significantly overestimates the weight on all minority classes, which greatly impacts the estimate of the majority class. DP-SYN did not capture an existence of any minority classes.

A standard measure for diversity between the original distribution P and synthetic distribution Q includes Kullback-Leibler (KL) divergence $D_{KL}(P||Q)$. Under differential privacy the support of P is a private information, so the private synthetic data inherently cannot ensure its support to align with the original data. This makes $D_{KL}(P||Q)$ and $D_{KL}(Q||P)$ (and related metrics such as Inception score [27]) undefined. One alternative is Jensen–Shannon divergence (JSD) [24,54]: $\text{JSD}(P||Q) := \frac{1}{2}D_{KL}(P||Q) + \frac{1}{2}D_{KL}(Q||P)$ which is always defined and nonnegative. We use this metric to evaluate the diversity of the synthetic data.

In addition, we propose another diversity measure, *μ-smoothed Kullback-Leibler (KL) divergence* between the original distribution P and synthetic distribution Q:

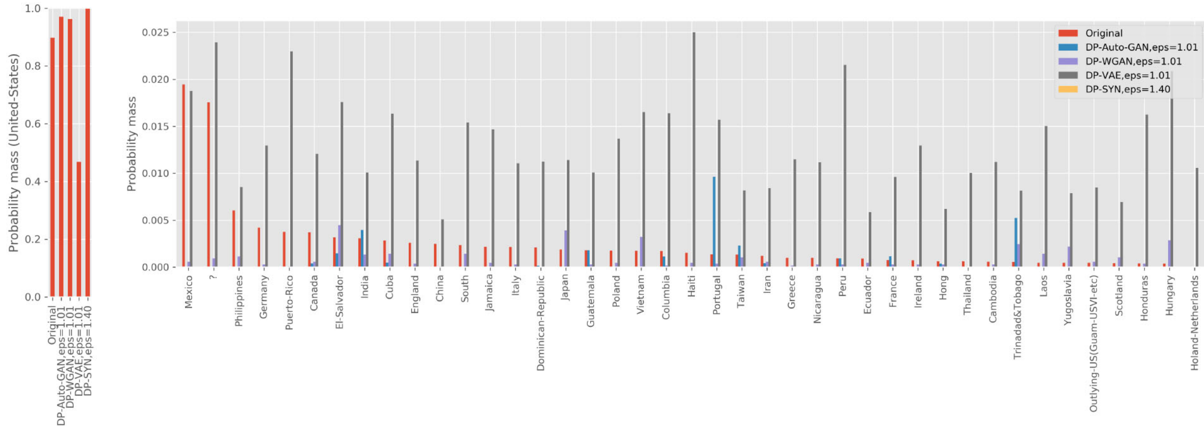
$$D^{\mu}_{KL}(P||Q) := \sum_{x \in supp(P)} (P(x) + \mu) \log\left(\frac{P(x)+\mu}{Q(x)+\mu}\right),$$

for small $\mu > 0$. D^{μ}_{KL} maintains the desirable property that $D^{\mu}_{KL} \geqslant 0$ and is zero if and only if $P = Q$. Smaller μ implies stronger penalties for missing minority categories in the synthetic data, and the penalty approaches ∞ as $\mu \to 0$. This allows μ as a knob to adjust the penalty necessary in private setting.



(a) Marital-status (b) Race

Fig. 9. Histograms of synthetic data generated by different algorithms.



(a) US (b) Minority classes

Fig. 10. Histogram of native-country features of the original data and synthetic data generated by different algorithms.

In our settings, we are concerned with one category dominating in the original distribution P (e.g., as in Fig. 9), say $P = (p_1, \ldots, p_k)$ with high $p_1 = \max_i p_i$, and when the synthetic distribution $Q = (q_1, \ldots, q_k) = (1, 0, \ldots, 0)$ supports only one single category. Then, we have $D^{\mu}_{KL}(P||Q) = \sum_{i=2}^{k}(p_i + \mu)\log(p_i + \mu) - (p_1 + \mu)\log(\frac{p_1+\mu}{1+\mu}) - (\sum_{i=2}^{k}(p_i + \mu))\log\mu$. For small $\mu > 0$, $\log\mu$ dominates $\log(p_i + \mu)$ and $\log(\frac{p_1+\mu}{1+\mu})$, so the dominating term is $\left(\sum_{i=2}^{k}(p_i + \mu)\right)\log\mu \approx (1 - p_1)\log\mu$. Hence, we use $\mu = e^{-\frac{1}{1-p_1}}$ so that this term is a constant, thus normalizing scores across features.

Table 3 reports the diversity divergences of all four algorithms for marital-status, race, and the sum across eight categorical features. One out of the nine categorical features are not used due to a difference in pre-processing of DP-SYN; see Appendix B for details. Both measures are able to detect the lost of diversity in DP-SYN in race, and identify DP-auto-GAN as generating more diverse data than the prior methods for most features and ϵ values.

We note that predictive scores may also not be appropriate for continuous features when no good classifier exists to predict the feature, even in the original dataset. In our setting, we found three continuous features with R^2 scores close to zero even with more complex regression models, and with negative R^2 scores on synthetic data, which is not meaningful. For those features, 1-way marginals (histograms, explored next) are preferred to prediction scores.

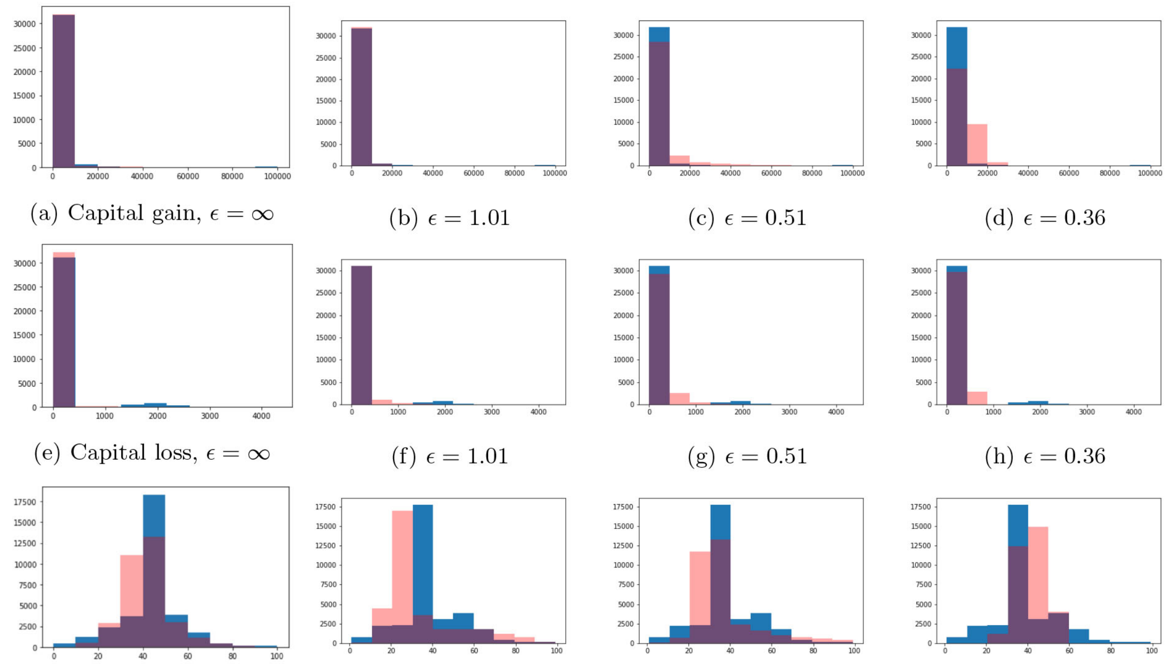
In general, we suggest that an evaluation of synthetic data should be based on probability measures (distributions of data) and not predictive scores of models. Models may be a source of not only unpredictability and instability, but also of bias and unfairness.

Histograms for continuous features with small r^2 scores. For three continuous features in the ADULT dataset (capital gain, capital loss, and hours worked per week), we were not able to find a regression model with

Table 3

Diversity measures JSD and D^{μ}_{KL} on different features of ADULT data and the sum of divergences across all eight applicable categorical features (All). Recall that p_1 is the maximum probability across all categories of that feature in the original data. Smaller values for the diversity measures imply more diverse synthetic data. For each row (feature), the smallest value for each setting of ϵ is highlighted in bold

	DP-auto-GAN			DP-WGAN			DP-VAE			DP-SYN		
ϵ values	0.36	0.51	1.01	0.36	0.51	1.01	0.36	0.51	1.01	0.50	0.80	1.40
					JSD diversity measure							
Marital	0.025	0.043	**0.014**	0.119	0.624	0.136	0.139	0.043	0.021	**0.017**	**0.013**	0.017
Race	**0.021**	**0.014**	0.016	0.081	0.053	0.040	0.095	0.031	**0.011**	0.053	0.053	0.053
All	0.33	**0.23**	**0.19**	1.29	2.41	0.73	0.80	0.44	0.23	**0.25**	0.27	0.28
					D^{μ}_{KL} Diversity Measure, with $\mu = e^{-\frac{1}{1-p_1}}$							
Marital	0.019	0.053	**0.005**	0.165	1.16	0.290	0.207	0.044	0.017	**0.017**	**0.011**	0.012
Race	**0.125**	**0.064**	0.089	0.262	0.465	0.277	0.315	0.102	**0.038**	0.465	0.465	0.465
All	**0.81**	**0.48**	**0.53**	5.26	6.39	1.53	2.52	1.17	0.58	0.99	1.00	1.02



(a) Capital gain, $\epsilon = \infty$ (b) $\epsilon = 1.01$ (c) $\epsilon = 0.51$ (d) $\epsilon = 0.36$

(e) Capital loss, $\epsilon = \infty$ (f) $\epsilon = 1.01$ (g) $\epsilon = 0.51$ (h) $\epsilon = 0.36$

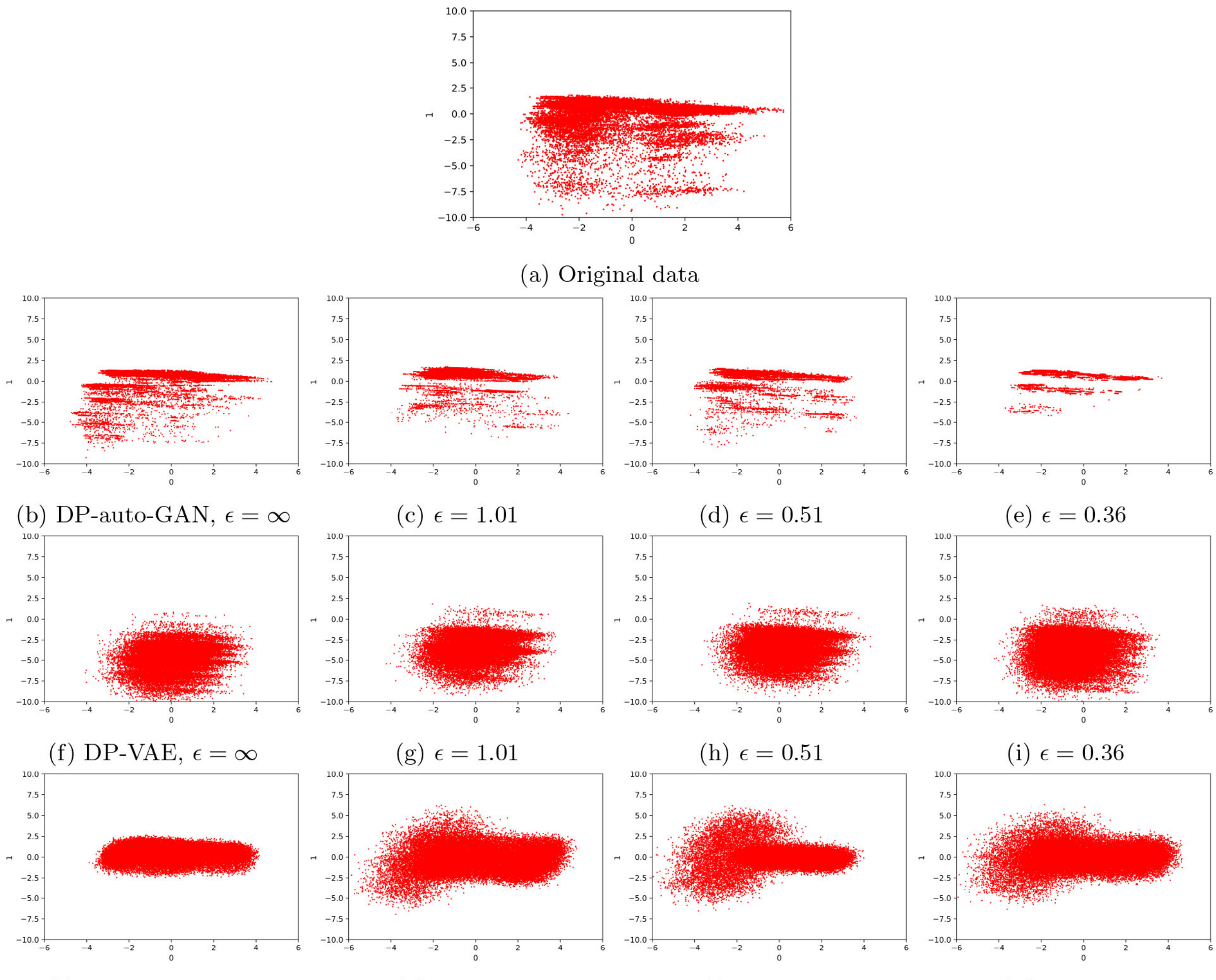
(i) Hours/week, $\epsilon = \infty$ (j) $\epsilon = 1.01$ (k) $\epsilon = 0.51$ (l) $\epsilon = 0.36$

Fig. 11. 1-way histogram for different values of ϵ. Each pair of consecutive rows correspond to capital gain, capital loss and weekly work-hours, respectively. Blue corresponds to the histogram of the real dataset, and red corresponds to the histogram of the synthetic dataset generated by DP-auto-GAN with the indicated ϵ. The overlap of both histograms is purple.

good fit (as measured by R^2 score) for the latter three features (capital gain, capital loss, and hours worked per week) in terms of the other features even on the real data. We attempted several different approaches, ranging from simple regression models such as lasso to complex models such as neural networks, and all had a low R^2 score on both the real and synthetic data. The capital gain and capital loss attributes are inherently hard to predict because the data are sparse (mostly zero) in these attributes.

Since the R^2 scores did not prove to be a good metric for these features, we instead plotted 1-way feature marginal histograms for each of these three remaining features to check whether the marginal distribution was learned correctly. These 1-way histograms for DP-auto-GAN are shown in Fig. 11. The figure shows that DP-auto-GAN identifies the marginal distribution of capital gain and capital loss quite well, and it does reasonably well on the hours-per-week feature.

Random forest prediction scores. Following [14],



(a) Original data

(b) DP-auto-GAN, $\epsilon = \infty$ (c) $\epsilon = 1.01$ (d) $\epsilon = 0.51$ (e) $\epsilon = 0.36$

(f) DP-VAE, $\epsilon = \infty$ (g) $\epsilon = 1.01$ (h) $\epsilon = 0.51$ (i) $\epsilon = 0.36$

(j) DP-SYN, $\epsilon = \infty$ (k) $\epsilon = 1.40$ (l) $\epsilon = 0.80$ (m) $\epsilon = 0.50$

Fig. 12. Scatterplots of projection of ADULT original and synthetic datasets on first two principle components of the original dataset. Synthetic datasets are generated from several algorithms at different ϵ values.

we also evaluate the quality of synthetic data by the accuracy of a random forest classifier to predict the label “salary” feature. In particular, we train a random forest classifier on synthetic data and test on the holdout original data, and report the F_1 accuracy score. The aim is that a classifier trained on synthetic data should report a similar accuracy score as the one trained on the original data.

In Table 3, we report the accuracy of synthetic datasets generated by DP-auto-GAN and DP-WGAN [14]. The results reported in [14] use $\epsilon = 3, 7, \infty$, whereas our algorithms used parameter values $\epsilon = 0.36, 0.51, 1.01, \infty$, a significant improvement in privacy. We see that our accuracy guarantees are higher than those of [14] with smaller ϵ values, and DP-auto-GAN achieved higher accuracy in the non-private setting. We note that part of the accuracy discrepancy because DP-auto-GAN can handle mixed-typed features, whereas DP-WGAN only handles categorical features.

2-Way PCA. In order to understand combined qualitative performance of all features, we show 2-way PCA marginal in Fig. 12. We fix the same projection from the original data, and require the synthetic data to be of the same format and go through the same preprocessing. For this reason, we do not compare to DP-WGAN since the original implementation [14] does not handle continuous columns, and we apply our preprocessing rather than the original preprocessing for DP-SYN. A qualitative inspection of the plots clearly shows the similarities of trends between the plots for real dataset and synthetic data generated by DP-auto-GAN for different values of ϵ, as low as $\epsilon = 0.51$.

Figure 12 is able to depict a qualitative description of the synthetic datasets that dimension-wise probability and predictive scores may not capture. DP-auto-GAN is able to capture the overall structure of 2-dimension PCA, with more points collapsing into a cluster as privacy budget ϵ decreases. The algorithm's internal GAN structure, however, is able to generate points on different clusters even at smaller ϵ, which better matches the projection of the original dataset. DP-VAE has a clear 1-cluster Gaussian-like distribution of 2-way PCA, which is consistent with this method's assumption that the underlying distribution in the latent space is Gaussian. The autoencoder transforms some datapoints near the edge of the cluster to shapes similar to the original 2-way PCA, but most datapoints remain at the center of the cluster. DP-SYN is able to capture the large single cluster in the original data, but is not diverse enough to capture small clusters. This is consistent with our previous observations on diversity under DP-SYN and the fact that DP-SYN assumes a mixture of Gaussian distributions in latent space, which may not be diverse or complex enough to capture smaller clusters of the original distribution.

DP-SYN and DP-VAE do not improve 2-way PCA plots as ϵ increases, suggesting that the underlying assumptions in latent space are likely a bottleneck. Interestingly, for $\epsilon = \infty$, DP-SYN synthetic data collapse to a single flat cluster (it is possible that many clusters are generated, but only one appears due to PCA), suggesting that DP-SYN overfits to the majority, and that adding noise for privacy splits the cluster. DP-auto-GAN 2-way PCA does not show structural limitations, but rather a promising result that it is possible to generate more detailed distributions that are closer to the original data.

6. Conclusion

We propose DP-auto-GAN – a combination of DP-autoencoder and DP-GAN – for differentially private data generation of mixed-type data. The inclusion of the autoencoder improves the efficacy of GANs, especially for high-dimensional data. Our method enjoys a 5x privacy improvement compared to [14] on the ADULT dataset in 14 dimensions and greater 100x improvement compared to [15] on a higher 1071-dimensional dataset, and achieves a meaningful privacy $\epsilon < 1$ for practical use. This approach is more complex than assuming a standard Gaussian distribution as in DP-VAE [16], and is better able to learn relationships among features.

Acknowledgments

Uthaipon Tantipongpipat was supported in part by NSF grants AF-1910423 and AF-1717947. Chris Waites was supported in part by a President's Undergraduate Research Award from the Georgia Institute of Technology. Digvijay Boob was supported in part by NSF grant CCF-1909298. Rachel Cummings was supported in part by a Mozilla Research Grant, a Google Research Fellowship, a JPMorgan Chase Faculty Award, and NSF grants CNS-1850187 and CNS-1942772. Part of this work was completed while Rachel Cummings was visiting the Simons Institute for the Theory of Computing. Most of this work was completed while all authors were affiliated with the Georgia Institute of Technology.

References

[1] Narayanan A, Shmatikov V. Robust De-anonymization of Large Sparse Datasets. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy. Oakland S&P '08; 2008. pp. 111-125.

[2] Barbaro M, Zeller T. A Face is Exposed for AOL Searcher No. 4417749. New York Times; 2006. [Online, Retrieved 9/25/2019]. New York Times. Available from: https//www.nytimes.com/2006/08/09/technology/09aol.html.

[3] Ohm P. Broken promises of privacy: Responding to the surprising failure of anonymization. UCLA Law Review. 2010; 57: 1701-1777.

[4] Carlini N, Liu C, Erlingsson Ú, Kos J, Song D. The Secret Sharer: Evaluating and testing unintended memorization in neural networks. In: Proceedings of the 28th USENIX Security Symposium. USENIX Security '19; 2019. pp. 267-284.

[5] Dwork C, McSherry F, Nissim K, Smith A. Calibrating noise to sensitivity in private data analysis. In: Proceedings of the 3rd Conference on Theory of Cryptography. TCC '06; 2006. pp. 265-284.

[6] Triastcyn A, Faltings B. Generating artificial data for private deep learning. In: Proceedings of the PAL: Privacy-Enhancing Artificial Intelligence and Language Technologies. PAL '18; 2018. pp. 33-40.

[7] Blum A, Ligett K, Roth A. A learning theory approach to non-interactive database privacy. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing. STOC '08; 2008. pp. 609-618.

[8] Hardt M, Rothblum GN. A multiplicative weights mechanism for privacy-preserving data analysis. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science. FOCS '10; 2010. pp. 61-70.

[9] Kingma DP, Welling M. Auto-encoding variational bayes; 2013. ArXiv preprint 1312.6114.

[10] Abay NC, Zhou Y, Kantarcioglu M, Thuraisingham B, Sweeney L. Privacy preserving synthetic data release using deep learning. In: Machine Learning and Knowledge Discovery in Databases (ECML PKDD '18). vol. 11051 of Lecture Notes in Computer Science. Springer; 2018. pp. 510-526.

[11] Chen Q, Xiang C, Xue M, Li B, Borisov N, Kaarfar D, et al. Differentially Private Data Generative Models; 2018. ArXiv preprint 1812.02274.

[12] Johnson AEW, Pollard TJ, Shen L, Li-wei HL, Feng M, Ghassemi M, et al. MIMIC-III, a freely accessible critical care database. Scientific Data. 2016; 3: 160035.

[13] Dua D, Graff C. UCI Machine Learning Repository; 2017. Available from: http://archive.ics.uci.edu/ml.

[14] Frigerio L, de Oliveira AS, Gomez L, Duverger P. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In: International Conference on ICT Systems Security and Privacy Protection. IFIP SEC '19 2019. pp. 151-164.

[15] Xie L, Lin K, Wang S, Wang F, Zhou J. Differentially private generative adversarial network; 2018. ArXiv preprint 1802.06739.

[16] Acs G, Melis L, Castelluccia C, De Cristofaro E. Differentially private mixture of generative neural networks. IEEE Transactions on Knowledge and Data Engineering. 2018; 31(6): 1109-1121.

[17] Hardt M, Ligett K, McSherry F. A simple and practical algorithm for differentially private data release. In: Advances in Neural Information Processing Systems 25, NIPS '12; 2012. pp. 2339-2347.

[18] Gaboardi M, Arias EJG, Hsu J, Roth A, Wu ZS. Dual query: Practical private query release for high dimensional data. In: Proceedings of the 31st International Conference on Machine Learning. ICML '14; 2014. pp. 1170-1178.

[19] Zhang J, Cormode G, Procopiuc CM, Srivastava D, Xiao X. PrivBayes: Private data release via Bayesian networks. ACM Transactions on Database Systems (TODS). 2017; 42(4): 25.

[20] Ping H, Stoyanovich J, Howe B. DataSynthesizer: Privacy-preserving synthetic datasets. In: Proceedings of the 29th International Conference on Scientific and Statistical Database Management. SSDBM '17; 2017. pp. 421-42:5.

[21] Surendra H, Mohan HS. A review of synthetic data generation methods for privacy preserving data publishing. International Journal of Scientific and Technology. 2017; 6: 95-101.

[22] Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, et al. Deep learning with differential privacy. In: Proceedings of the 2016 ACM Conference on Computer and Communications Security. CCS '16; 2016. pp. 308-318.

[23] Mironov I. Rényi differential privacy. In: Proceedings of the 2017 IEEE 30th Computer Security Foundations Symposium. CSF '17; 2017. pp. 263-275.

[24] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Nets. In: Advances in Neural Information Processing Systems 27, NIPS '14, 2014. pp. 2672-2680.

[25] Mogren O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. Constructive Machine Learning Workshop (CML) at NeurIPS 2016, 2016.

[26] Saito M, Matsumoto E, Saito S. Temporal generative adversarial nets with singular value clipping. In: Proceedings of the IEEE International Conference on Computer Vision. ICCV '17, 2017. pp. 2830-2839.

[27] Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X. Improved Techniques for Training GANs. In: Advances in Neural Information Processing Systems 29, NIPS '16, 2016. pp. 2234-2242.

[28] Jang E, Gu S, Poole B. Categorical reparameterization with Gumbel-softmax. In: Proceedings of the 5th International Conference on Learning Representations. ICLR '17; 2017. Available from: https//openreview.net/forum?id=rkE3y85ee.

[29] Kusner MJ, Hernández-Lobato JM. GANs for sequences of discrete elements with the Gumbel-softmax distribution; 2016. ArXiv preprint 1611.04051.

[30] Wang H, Wang J, Wang J, Zhao M, Zhang W, Zhang F, et al. GraphGAN: Graph representation learning with generative adversarial nets. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence. AAAI '18; 2018. pp. 2508-2515.

[31] Xu L, Skoularidou M, Cuesta-Infante A, Veeramachaneni K. Modeling tabular data using Conditional GAN. In: Advances in Neural Information Processing Systems 32, NeurIPS '19; 2019. pp. 7333-7343.

[32] Lim SK, Loo Y, Tran NT, Cheung NM, Roig G, Elovici Y. DOPING: Generative data augmentation for unsupervised anomaly detection with GAN. In: Proceedings of the 2018 IEEE International Conference on Data Mining. ICDM '18; 2018. pp. 1122-1127.

[33] Park N, Mohammadi M, Gorde K, Jajodia S, Park H, Kim Y. Data synthesis based on generative adversarial networks. Proceedings of the VLDB Endowment. 2018; 11(10): 1071-1083.

[34] Arjovsky M, Chintala S, Bottou L. Wasserstein GAN, 2017. ArXiv preprint 1701.07875.

[35] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC. Improved Training of Wasserstein GANs. In: Advances in Neural Information Processing Systems 30, NIPS '17; 2017. pp. 5767-5777.

[36] Alzantot M, Srivastava M. Differential Privacy Synthetic Data Generation using WGANs, 2019. Available from: https://github.com/nesl/nist_differential_privacy_synthetic_data_challenge/.

[37] Mirza M, Osindero S. Conditional generative adversarial nets, 2014. ArXiv preprint 1411.1784.

[38] Torkzadehmahani R, Kairouz P, Paten B. DP-CGAN: Differentially Private Synthetic Data and Label Generation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2019.

[39] Papernot N, Abadi M, Erlingsson U, Goodfellow I, Talwar K. Semi-supervised knowledge transfer for deep learning from private training data. In: International Conference on Learning Representations. ICLR '17, 2017. Available from: https//openreview.net/forum?id=HkwoSDPgg.

[40] Papernot N, Song S, Mironov I, Raghunathan A, Talwar K, Erlingsson Ú. Scalable private learning with PATE. In: International Conference on Learning Representations. ICLR '18, 2018. Available from: https//openreview.net/forum?id=rkZB1XbRZ.

[41] Jordon J, Yoon J, van der Schaar M. PATE-GAN: generating synthetic data with differential privacy guarantees. In: Proceedings of the 7th International Conference on Learning Representations. ICLR '19; 2019. Available from: https//openreview.net/forum?id=S1zk9iRqF7.

[42] Park M, Foulds J, Choudhary K, Welling M. DP-EM: Differentially Private Expectation Maximization. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. AISTATS '17; 2017. pp. 896-904.

[43] Zhang X, Ji S, Wang T. Differentially Private Releasing via Deep Generative Model (Technical Report); 2018. ArXiv preprint1801.01594.

[44] NIST. Contest: NIST DIfferential Privacy #3; 2019. National Institute of Standards and Technology, Public Safety Communications Research. TopCoder. Available from: https//community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=17421&pm=15315.

[45] NIST. Differential Privacy Synthetic Data Challenge Algo-

rithms; 2019. National Institute of Standards and Technology, Privacy Engineering Program. NIST Information Technology Laboratory/Applied Cybersecurity Division. Available from: https://www.nist.gov/itl/applied-cybersecurity/privacy-engineering/collaboration-space/browse/de-identification-tools#dpchallenge.

[46] Charest AS. How can we analyze differentially-private synthetic datasets? Journal of Privacy and Confidentiality. 2011; 2(2): 21-33.

[47] Choi E, Biswal S, Malin B, Duke J, Stewart WF, Sun J. Generating multi-label discrete patient records using generative adversarial networks. In: Proceedings of Machine Learning for Healthcare, 2017, pp. 286-305.

[48] McMahan HB, Andrew G. A general approach to adding differential privacy to iterative training procedures. PPML18: Privacy Preserving Machine Learning – NeurIPS 2018 Workshop. 2018.

[49] Google. TensorFlow Privacy; 2018. Available from: https://github.com/tensorflow/privacy.

[50] Van Erven T, Harremos P. Rényi divergence and Kullback-Leibler divergence. IEEE Transactions on Information Theory. 2014; 60(7): 3797-3820.

[51] Wang YX, Balle B, Kasiviswanathan S. Subsampled Rényi Differential Privacy and Analytical Moments Accountant. In: Proceedings of the 22th International Conference on Artificial Intelligence and Statistics. AISTATS '19; 2019. pp. 1226-1235.

[52] Borji A. Pros and cons of GAN evaluation measures. Computer Vision and Image Understanding. 2019; 179: 41-65.

[53] Bagdasaryan E, Poursaeed O, Shmatikov V. Differential privacy has disparate impact on model accuracy. In: Advances in Neural Information Processing Systems 32, NeurIPS '19; 2019. pp. 15479-15488.

[54] Jeffreys H. An invariant form for the prior probability in estimation problems. Proceedings of the Royal Society of London Series A Mathematical and Physical Sciences. 1946; 186(1007): 453-461.

A. Training details for experiments on MIMIC-III data

MIMIC-III data set contains 46,520 data points in total, and is partitioned into train, validation, and test data sets of sizes 27912, 9304, 9304 (60%, 20%, 20%), respectively. Privacy analysis is calculated using the training size. Data are stored in 0/1 format. DP-auto-GAN pre-trained models in this paper are available at https://github.com/DPautoGAN/DPautoGAN/tree/master/results/pre-trained%20models.

DP-auto-GAN Computing Infrastructure. DP-auto-GAN are run on GCP: n1-highmem-2 (2 vCPUs, 13 GB memory) with 1 x NVIDIA Tesla K80. The training of autoencoder (for 15,000 iterations) and of GAN (for 20,000 iterations) each takes about 1.5 hours. The combined training together with performance evaluations (probability and prediction plots) are done in less than 4 hours for each setting of parameter.

DP-auto-GAN Training. The autoencoder was trained via Adam with Beta $1 = 0.9$, Beta $2 = 0.999$, and a learning rate of 0.001. It was trained on minibatches of size 100 and microbatches of size 1. L2 clipping norm was selected to be the median L2 norm observed in a non-private training loop, set to 0.8157. The noise multiplier was then calibrated to achieve the desired privacy guarantee.

The GAN was composed of two neural networks, the generator and the discriminator. The generator was a simple feed-forward neural network, trained via RMSProp with alpha $= 0.99$ with a learning rate of 0.001. The discriminator was also a simple feed-forward neural network, also trained via RMSProp with the same parameters, with minibatches of size 128. The L2 clipping norm of the discriminator was set to 0.35. The pair was trained on minibatches of size 1,000 and a microbatch size of 1, with 2 updates to the discriminator per 1 update to the generator. Again, the noise multiplier was then calibrated to achieve desired privacy guarantees.

Selecting the Noise Multipliers and the Numbers of Iterations. Noise multipliers are finally set at $\psi = 3.5$, 2.3, 1.3 simultaneously to both autoencoder and GAN to achieve $\epsilon = 0.81$, 1.33, 2.70 respectively. Training is first done for 20000 iterations, and the generated data every 1000 iterations are saved. We then inspect whether an earlier trained model may be used as follows. For $\psi = 2.3$, 1.3, the number of features where the model outputs all zero converges to 181 out of 1071 features and stabilize at 181 for the remaining of training. We picked the second saved model which has 181 such features. For $\psi = 3.5$the number of such features quickly drops to 181–182 and then fluctuates between 181–182 in the remaining of the training. We pick the third saved model which has the number of such features being 181 or 182. The final iterations picked is $T = 6000$, 7000, 7000 for $\psi = 3.5$, 2.3, 1.3 respectively, and these numbers of iterations are then used to calculate the privacy parameters.

Model Architecture. A serialization of the (non-private, i.e. $\epsilon = \infty$) model architectures used in the experiment can be found below. For the private version, we change the latent dimension from 128 to 64.

```
(encoder): Sequential(
(0): Linear(in-feature=1071, out-feature=128, bias=True)
(1): Tanh()
)
(decoder): Sequential(
(0): Linear(in-feature=128, out-feature=1071, bias=True)
(1): Sigmoid()
)

Generator(
(model): Sequential(
(0): Linear(in-feature=128, out-feature=128)
(1): LeakyReLU(negative-slope=0.2)
(2): Linear(in-feature=128, out-feature=128)
(3): Tanh()
)
)
Discriminator(
(model): Sequential(
(0): Linear(in-feature=1071, out-feature=256, bias=True)
```

(1): LeakyReLU(negative-slope=0.2)
(2): Linear(in-feature=256, out-feature=1, bias=True))
)

B. Training details for experiments on ADULT data

In this section, we describe experimental details of DP-auto-GAN, DP-WGAN, DP-VAE, and DP-SYN for reproducibility.

For DP-auto-GAN and in original implementation of existing algorithms, data are preprocessed by one-hot encoding categorical features, and by max-min scalar on continuous features, i.e. mapping maximum to 1 and minimum to 0. While maximum and minimum are technically leaking privacy, and hence not public as assumed in our framework, they are sometimes treated as publicly available such as in synthetic data challenge [44] and assumed in implementation of existing works. In other usage, reasonable cap can be assumed on features, or a standard differentially private query on minimum and maximum of a feature on a bounded range can be used.

Synthetic data are generated from each trained generative model to a size of 32561, the training size (which is two thirds) of ADULT data. (Results of dimension-wise prediction we observed are similar with synthetic data of full ADULT size 48842).

In parameter tuning of pre-existing methods from original authors, we keep the whole framework including preprocessing of ADULT data, architecture, optimizer, and other hyper-parameters, except the noise multiplier. We attempted to keep original architectures as they are likely optimized in original work for generating synthetic ADULT data. We tune the noise multiplier on several higher values to achieve a smaller ϵ values needed, and pick the est performing model across noise multiplier values used. Details for each algorithm can be found in the remainder of this section.

Computing infrastructure and runtime. DP-auto-GAN are run on GCP: n1-highmem-2 (2 vCPUs, 13 GB memory) with 1 x NVIDIA Tesla K80. The combined training on autoencoder and GAN are done in approximately 2–3 hours for each setting of parameter. DP-WGAN, DP-VAE, DP-SYN are run on a personal computer with processor Intel(R) Core(TM) i7-6600U CPU @ 2.60 GHz 2.81 GHz and RAM 16.0 GB. Training of DP-WGAN, DP-VAE, DP-SYN for each parameter setting finishes in between approximately 15 minutes to 2 hours. Any of all evaluation metrics to an ADULT synthetic dataset finishes in less than 1–2 minutes.

B.1. DP-auto-GAN training

Preprocessing. Original ADULT dataset contains 15 features, one of which is a positive integer feature named "fnlwgt" (final weight). This feature is discarded as unrelated to each individual person in the census, but rather the additional feature US census created by mapping a person to an estimated weight of another demographic dataset. Two features "education" and "education-num" are the same feature representing in a different format – string or a positive integer. In particular, education consists of 16 levels of education, and education-num represents them as numbers 1, 2, . . .,16. Hence, we remove one of these two features and treat this column as one categorical feature. In the end, we have 9 categorical features, one of which is a binary label named "salary", and four continuous features.

ADULT data consists of 48842 datapoints, partitioned into 32561 (two-thirds) for training and 16281 (one-third) for testing. We follow the same partitioning by training our DP-auto-GAN on 32561 samples and holding the rest only for synthetic data evaluation.

Training. The autoencoder was trained via Adam with Beta 1 = 0.9, Beta 2 = 0.999, and a learning rate of 0.005 for 10,000 minibatches of size 64 and a microbatch size of 1. The L2 clipping norm was selected to be the median L2 norm observed in a non-private training loop, equal to 0.012. The noise multiplier was then calibrated to achieve the desired privacy guarantee. The final noise multiplier used for $\epsilon = 0.36, 0.51, 1.01$ are $\psi = 5, 2.5, 1.5$, respectively.

The GAN was composed of two neural networks, the generator and the discriminator. The generator used a ResNet architecture, adding the output of each block to the output of the following block. It was trained via RMSProp with alpha = 0.99 with a learning rate of 0.005. The discriminator was a simple feed-forward neural network with LeakyReLU hidden activation functions, also trained via RMSProp with alpha = 0.99. The L2 clipping norm of the discriminator was set to 0.022. The pair was trained on 15,000 minibatches of size 128 and a microbatch size of 1, with 15 updates to the discriminator per 1 update to the generator. Again, the noise multiplier was then calibrated to achieve the desired privacy guarantee. The final noise multiplier used for $\epsilon = 0.36, 0.51, 1.01$ are $\psi = 8, 7.5, 3.5$, respectively.

Model architecture. A serialization of the model architectures used in the experiment can be found below. Note that the number of latent dimension, 64, is the same as in the implementation of DP-SYN.

```
Autoencoder(
(encoder): Sequential(
0: Linear(in-features=106, out-feature=60, bias=True)
(1): LeakyReLU(negative-slope=0.2)
(2): Linear(in-feature=60, out-feature=15, bias=True)
(3): LeakyReLU(negative-slope=0.2)
)
(decoder): Sequential(
(0): Linear(in-feature=15, out-feature=60, bias=True)
(1): LeakyReLU(negative-slope=0.2)
(2): Linear(in-feature=60, out-feature=106, bias=True)
(3): Sigmoid()
)
)
Generator(
(block-0): Sequential(
(0): Linear(in-feature=64, out-feature=64, bias=False)
(1): BatchNorm1d()
(2): LeakyReLU(negative-slope=0.2)
)
(block-1): Sequential(
(0): Linear(in-feature=64, out-feature=64, bias=False)
(1): BatchNorm1d()
(2): LeakyReLU(negative-slope=0.2)
)
(block-2): Sequential(
(0): Linear(in-feature=64, out-feature=15, bias=False)
(1): BatchNorm1d()
(2): LeakyReLU(negative-slope=0.2)
)
)
Discriminator(
(model): Sequential(
(0): Linear(in-feature=106, out-feature=70, bias=True)
(1): LeakyReLU(negative-slope=0.2)
(2): Linear(in-feature=70, out-feature=35, bias=True)
(3): LeakyReLU(negative-slope=0.2)
(4): Linear(in-feature=35, out-feature=1, bias=True) )
)
```

B.2. DP-WGAN training

Preprocessing. The algorithm of WGAN [14] (their implementation can be found at https://github.com/SAP-samples/security-research-differentially-private-generative-models) is used and implemented only for discrete data. The preprocessing automatically delete continuous columns. Hence, DP-WGAN preprocesses ADULT data into 9 categorical features, one of which is the binary "salary" label. Each categorical feature is then one-hot encoded before feeding into DP-WGAN training.

Parameter tuning. The original implementation uses noise multiplier $\psi = 7$, originally for a higher values of epsilons as mentioned in [14]. We found that the training cannot achieve $\epsilon = 0.51$ just after one epoch, while the typical training requires multiple (tens to almost a hundred) epochs. Hence, we train with higher noise parameters $\psi = 7, 9, 11, 13, 15, 19, 23, 27.5, 35$. For $\epsilon = 0.36$, we also attempted $\psi = 40, 45, 50, 60, 70, 80, 100$ which still gives $\epsilon > 0.36$ only after one epoch. However, if we relax to $\epsilon = 0.37$, we are able to train a few epochs at high noise level to achieve the privacy guarantee, so we allow $\epsilon = 0.37$.

For each of the noise parameter, we select the generated data from the epoch before privacy busget is exhausted. The synthetic data is evaluated by dimension-wise prediction. We exclude a few cases where prediction score of the salary feature is zero, indicating possibly a mode collapse. Then, we pick the model for each ϵ setting across different noise multiplier with highest total prediction score. We note that overall performance of DP-WGAN are comparable across $\psi \in \{7, 9, 11, 13\}$ for $\epsilon \geqslant 0.8$ and less predictable for smaller ϵ. The final ψ we use for $\epsilon = 0.36, 0.51, 1.01$ are $\psi = 27.5, 19{,}9$, respectively. Dimension-wise prediction across different ϵ and ψ are available at https://github.com/DPautoGAN/DPautoGAN/tree/master/results/prediction-plots/DP-WGAN.

B.3. DP-VAE training

Preprocessing. An implementation of DP-VAE (not by the author of original work [16]) can be found at https://github.com/SAP-samples/security-research-differentially-private-generative-models/blob/master/Tutorial_dp-VAE.ipynb for training on ADULT data. We slightly modify the size of the training set to 32561 sample points, as used in original dataset and our DP-auto-GAN training. Though the original implementation uses all 15 features, we preprocess the data exactly the same way as DP-auto-GAN preprocessing for a fairer reporting.

We, however, observed that dimension-wise prediction is either of similar overall quality when using original 15 dimensions instead of what is reported. Dimension-wise predictions of 13 and 15 features on several values of ϵ is available at https://github.com/DPautoGAN/DPautoGAN/tree/master/results/prediction-plots/DP-VAE/vae-13-vs-15-features.

Parameter tuning. The tutorial defaults noise multiplier at $\psi = 1$. To get smaller ϵ, we test $\psi = 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5$ and additionally $\psi = 5.5, 6, 6.5, 7, 7.5, 8$ for $\epsilon = 0.36$. Standard validation accuracy score of VAE in the training process from the keras package are used. We observed an expected pattern that accuracy increases from very small ψ until it drops again at some high ψ value, where the peak of ψ is larger for smaller ϵ. As a result, we extended ψ as mentioned for smaller ϵ to be certain that we have reached such peak. Then, the model from noise multiplier which gives highest accuracy score is used. The final ψ used for $\epsilon = 0.36, 0.51, 1.01$ are $\psi = 5, 4{,}2$, respectively.

B.4. DP-SYN training

Preprocessing. The original implementation of DP-SYN deletes "fnlwgt" and a redundant "education-num" as in our preprocessing. However, it group some similar educational levels into one category, resulting in 8 categories rather than 16. It preprocesses capital-gain and capital-loss into categorical features with 3 classes: "low", "medium", and "high." Hence, the final preprocessed data has 2 continuous and 11 categorical features. Because one categorical feature, education, is preprocessed differently from other algorithms, it is excluded from reporting the sum of diversity divergence scores in Table **??**.

When using the original preprocessing, we update the test set for dimension-wise prediction score to the same preprocessed format. We observed, however, that despite the architecture likely tuned for the original preprocessing, we ran DP-SYN using preprocessing and obtained a similar result. As we aim to report the optimized and most original version of previous work, we choose the original preprocessing in reporting results in this work. We suspect, however, that the empirical results and conclusions would be similar under our preprocessing.

Parameter tuning. Original ϵ and noise multiplier ψ used are $\psi = 2, 4$ for $\epsilon = 2.4, 3.2$ and $\psi = 4$ for $\epsilon = 1.6$ and $\psi = 4, 8$ for $\epsilon = 1.2$. We tested on $\psi = 2, 4, 6, 8$ for $\epsilon = 0.5, 0.8, 1.4$. $\psi = 2$ was not possible at $\epsilon = 0.5$ due to large ϵ incurred even within the first epoch of training. The implementation includes its own accuracy metric [10], which is an SVM classifier on the synthetic data. The model is trained 10 times for each noise and ϵ setting, and the noise which gives the highest average accuracy score for each ϵ setting is used. We saw a pattern of accuracy score either stay the same or increase from smallest ψ then later degrades, so we extended the range of ψ (up to 8 as mentioned) until we were certain that we have found a peak of accuracy score. We note that dimension-wise prediction performs similarly for $\psi = 2, 4, 6$ and slightly better at these ψ than the higher $\psi = 8$. Final ψ used for $\epsilon = 0.5, 0.8, 1.4$ is $\psi = 4$ for all three settings (and 6, 4, 4 if using our preprocessing). The results across all noise multiplier values can be found at https://github.com/DPautoGAN/DPautoGAN/tree/master/results/prediction-plots/DP-SYN.

DP-SYN first partitions the data into groups based on number of unique labels, which is 2 in "salary" label of the ADULT data. DP-EM then chooses the number of clusters in a mixture of Gaussian in each group by Calinski-Harabasz criterion. The range of numbers of clusters tested (which is also from the original implementation) is $K = 1, 2, \ldots, 7$.

Privacy accounting. We keep the original privacy accounting, which is to split ϵ, δ into halves, each for autoencoder and DP-EM, in the original implementation of DP-SYN. We allow higher ϵ which is computed by using standard composition instead of RDP composition on two phases of our DP-auto-GAN training. We also note that DP-SYN treats the label column as public, which is used in partitioning the original data into groups based on the label, whereas DP-auto-GAN, DP-WGAN, and DP-VAE treat all features as private.

The original implementation of DP-SYN was not able to finish a single epoch even for a large noise multiplier $\psi = 16,32$ to achieve $\epsilon = 0.5$ by a standard composition. We found that this is due to a loose analysis of RDP in the original implementation. We increase the moment order of 32 in the original implementation to 96 to obtain a tighter DP analysis, which allows the $\epsilon = 0.5$ (standard composition) results reported in this work.