Benchmarking Differentially Private Tabular Data Synthesis: [Experiments & Analysis]

KAI CHEN, University of Virginia, USA
XIAOCHEN LI, UNC Greensboro, USA
CHEN GONG, University of Virginia, USA
RYAN MCKENNA, Google Research, USA
TIANHAO WANG, University of Virginia, USA

Differentially private (DP) tabular data synthesis generates artificial data that preserves the statistical properties of private data while safeguarding individual privacy. The emergence of diverse algorithms in recent years has introduced challenges in practical applications, such as inconsistent data processing methods, the lack of in-depth algorithm analysis, and incomplete comparisons due to overlapping development timelines. These factors create significant obstacles to selecting appropriate algorithms.

In this paper, we address these challenges by proposing a benchmark for evaluating tabular data synthesis methods. We present a unified evaluation framework that integrates data preprocessing, feature selection, and synthesis modules, facilitating fair and comprehensive comparisons. Our evaluation reveals that a significant utility-efficiency trade-off exists among current state-of-the-art methods. Some statistical methods are superior in synthesis utility, but their efficiency is not as good as most deep learning-based methods. Furthermore, we conduct an in-depth analysis of each module with experimental validation, offering theoretical insights into the strengths and limitations of different strategies. Our code is open-sourced via the link.[1]

CCS Concepts: • **Security and privacy → Privacy-preserving protocols**.

Additional Key Words and Phrases: Differential Privacy, Tabular Data Synthesis

ACM Reference Format:
Kai Chen, Xiaochen Li, Chen Gong, Ryan McKenna, and Tianhao Wang. 2025. Benchmarking Differentially Private Tabular Data Synthesis: [Experiments & Analysis]. *Proc. ACM Manag. Data* 3, 6 (SIGMOD), Article 299 (December 2025), 25 pages. https://doi.org/10.1145/3769764

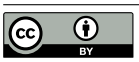
1 Introduction

Private tabular data synthesis generates artificial data that preserves the statistical properties of real data while protecting individual privacy. This critical problem has a broad range of applications in practice, extending from healthcare [30, 53] to governmental planning [5, 12] and beyond. For instance, in healthcare, there is a need to share patient data for medical treatment while preserving privacy. Similarly, in government, data analysts must analyze sensitive personal attributes while ensuring confidentiality. Addressing this challenge has thus received much attention.

Differential privacy (DP) has become the gold standard for protecting privacy. DP ensures that the inclusion or exclusion of any single data point does not significantly affect the outcome, thereby protecting each individual data point within a dataset. A substantial body of research has been

[1]https://github.com/KaiChen9909/tab_bench

Authors' Contact Information: Kai Chen, University of Virginia, USA, kaichen@virginia.edu; Xiaochen Li, UNC Greensboro, USA, X_LI12@uncg.edu; Chen Gong, University of Virginia, USA, chengong@virginia.edu; Ryan McKenna, Google Research, USA, mckennar@google.com; Tianhao Wang, University of Virginia, USA, tianhao@virginia.edu.



ACM 2836-6573/2025/12-ART299
https://doi.org/10.1145/3769764

proposed to address the data synthesis problem with DP. Based on their working principles, they can be broadly classified into two categories: statistical methods and deep learning methods. Statistical methods [24, 36, 39, 64, 65, 69, 69, 71] compress data information through statistical properties, such as low-dimensional data distributions, to achieve data generation. On the other hand, deep learning methods leverage deep learning frameworks designed for data generation, such as generative adversarial networks [21, 23, 28, 37, 61, 66] and diffusion models [31, 34, 48].

In addition to these efforts to address this problem, many works also focus on providing comprehensive benchmarks. For instance, Du et al. [16] and Tao et al. [60] try to evaluate current methods under the same setting. But their works do not involve recent algorithms and thus lack completeness. Some other benchmark works [17, 26, 67] focus more on making algorithm analysis and comparison, lacking necessary empirical validation.

In summary, even though several studies have been conducted in this field, we still face several challenges: (1) *Lack of unified evaluation settings.* Beyond algorithmic strategies, evaluation settings, such as data preprocessing, play a significant role in determining algorithm performance. However, these settings are often considered trivial and thus are frequently overlooked in many methods, which potentially leads to unfair comparisons between methods. (2) *Lack of systematic and in-depth analysis.* Current works often focus on proposing new methods, only providing limited intuition about algorithm analysis, or they only provide an overall comparison without detailed analysis, such as comparing deep learning and statistical methods' utility, but not delving into the reasons for the observed differences. Consequently, certain aspects of in-depth analysis, such as analysis for individual algorithm modules, remain underexplored. For example, questions such as how to select marginals more accurately are insufficiently addressed. However, many generative algorithms rely entirely on the selection of marginals, making this result particularly important. (3) *Lack of comprehensive comparison.* Due to various reasons, such as concurrent development or relatively recent introduction, comparisons between recently proposed methods and existing works remain incomplete, particularly between some representative methods like Private-GSD [36] and AIM [40]. Furthermore, current comparisons largely focus on the overall utility of algorithms, with little attention given to experiments analyzing specific working modules.

In light of the above challenges, we believe that proposing a new benchmark for evaluating tabular data synthesis is necessary. The contributions of our benchmark work are as follows.

Proposing a Unified Framework for Evaluation. We first propose a generalized framework and align all methods within this framework to ensure fair and objective comparisons. The framework consists of a *data preprocessing module*, a *feature selection module*, and a *data synthesis module*. Notably, this is the first framework to explicitly consider the impact of preprocessing on algorithm comparisons. Moreover, we move forward on the selection and synthesis modules by categorizing them according to their working principle, providing a new perspective to understand them better.

Providing Rigorous Analysis for Current Methods. Given our unified framework, we conduct an in-depth analysis of different modules. We consider different preprocessing methods and their drawbacks and advantages. For the feature selection module, we discuss the importance of introducing a scale penalty term during selection and prove the superiority of the adaptive marginal selection strategy. Finally, we investigate the efficiencies of current synthesis methods and discuss their potential limitations.

Conducting a Comprehensive Comparison. We include current state-of-the-art methods [7, 23, 37, 40, 64, 71] under both statistical and deep learning methods, and newly proposed methods [31, 36] that have not been thoroughly explored. Moreover, our evaluation is more fine-grained and helps us understand how each module of the algorithms functions independently.

We observed some important experimental findings, i.e., (1) A significant trade-off between utility and efficiency exists among current methods under current implementations. Two statistical methods, AIM [40] and PrivMRF [7], outperform in utility but show worse time efficiency. Deep learning methods, even though relatively inferior in utility, are highly efficient. (2) Preprocessing is crucial for improving algorithm efficiency without significant synthesis errors. It is also algorithm-dependent, with different techniques better suited to specific synthesis methods. (3) All current synthesis modules exhibit limitations in different ways, such as low efficiency or inferior utility.

2 Problem Formulation

Differential privacy (DP) has become the *de facto* standard for data privacy. It allows aggregated statistical information to be extracted while limiting the disclosure of information about individuals. More formally, the definition of DP is given by:

Definition 1 (Differential Privacy). *An algorithm $\mathcal{A}$ satisfies (ε, δ)-differential privacy ((ε, δ)-DP) if and only if for any two neighboring datasets D and D' and any $T \subseteq Range(\mathbf{A})$, we have*

$$\Pr \mathcal{A}(D) \in T \leq e^{\varepsilon} \Pr \mathcal{A}(D') \in T + \delta.$$

Here, we say two datasets are neighboring ($D \simeq D'$) when they differ on one tuple/sample. To achieve DP in different scenarios, many mechanisms have been employed, such as Gaussian mechanism [42], exponential mechanism [37, 40], and DP-SGD [43]. We provide a detailed introduction to them in the appendix of our full paper [10]. In our work, we use Rényi DP as a tight composition tool, defined as:

Definition 2 (Rényi DP [42]). *We say that an algorithm $\mathcal{A}$ satisfies (α, ε)-Rényi DP ((α, ε)-RDP) if and only if for any two neighboring datasets D and D'*

$$D_{\alpha}(\mathcal{A}(D)||\mathcal{A}(D')) \leq \varepsilon,$$

where $D_{\alpha}(Y||N) = \frac{1}{\alpha-1} \ln \mathbb{E}_{x\sim N} \left[\frac{Y(x)}{N(x)}\right]^{\alpha}$.

RDP has composition and post-processing properties [42], which make it a suitable choice for complex algorithm design. Moreover, an (α, ε)-RDP guarantee can easily be converted to a (ε', δ)-DP guarantee via Theorem 1 [42].

Theorem 1. *If f is an (α, ε)-RDP mechanism, then it also satisfy $\left(\varepsilon + \frac{\log 1/\delta}{\alpha-1}, \delta\right)$-DP for any $0 < \delta < 1$.*

One promising application of DP is for tabular data synthesis, wherein an artificial tabular dataset is generated that mirrors the statistical characteristics of the original dataset without compromising individual privacy. More formally, assuming that we have a dataset D composed of n records $\{x_1, \cdots, x_n\}$, and each record has d attributes $\{A_1, \cdots, A_d\}$, we want to generate a dataset D_s similar to D. The two datasets are considered similar based on some similarity metric, such as the ℓ_1 distance or the performance under a downstream task (e.g., answering a range query or training a classification task). We give more concrete metrics in Section 8.

3 Existing Work

This section examines existing DP tabular synthesis methods and identifies shortcomings in current benchmarks. These limitations inspire the development of our new benchmark.

Table 1. Summary of benchmarked algorithms. We match different algorithms within our framework and summarize/categorize their working pipelines. Here, 'unspecified' means that related information is not mentioned or does not have a deterministic setting in the original paper. The original baselines column lists some other well-received baseline algorithms.

Category	Method	Data Preprocessing	Feature Selection	Data Synthesis	Original Baselines
Statistical Method	RAP [3]	Unspecified	Adaptive	Relaxed Projection	FEM [65],HDMM [38]
	PrivSyn [71]	Categorical Preprocessing	Non-adaptive	GUM	PrivBayes [69], DualQuery [19],PGM [41]
	PrivMRF [7]	Unspecified	Adaptive	PGM	PrivBayes, BSG [6], DP-WGAN [55], DP-Copula [33]
	RAP++ [64]	Unspecified	Adaptive	Relaxed Projection	RAP, DP-MERF, DP-CTGAN [18], PGM
	AIM [40]	Unspecified	Adaptive	PGM	PrivMRF, RAP, MST [39], MWEM [24]
	Private-GSD [36]	Unspecified	Unspecified	Genetic Algorithm	GEM, RAP++, PGM [41]
deep learning Method	GEM [37]	Unspecified	Adaptive	Generative Network	RAP, MWEM, DualQuery
	DP-MERF [23]	Unspecified	Non-adaptive	Generative Network	DP-GAN [66], DP-CGAN [61]
	TabDDPM [31]	Unspecified	-	Diffusion Model	-

3.1 Existing Algorithms

Broadly, existing DP tabular data synthesis methods can be divided into two categories, which are statistical methods and deep learning methods.

Statistical Methods. The exploration of statistical methods started earlier. Shortly after the development of DP, researchers began investigating the data generation problem under the DP. MWEM [24] and DualQuery [19] both release data by repeatedly improving an approximated distribution using Multiplicative Weight approach [25]. Another notable line of research involves Bayesian networks, such as PrivBayes [69] and BSG [6]. In addition, Li et al. [33] try to utilize Copula functions for data generation.

In 2018 and 2020, NIST [46, 47] hosted challenges about DP data synthesis. Among the competing algorithms, PrivBayes, MST [39], and DPSyn [35] exhibited the best performance. All of these methods attempt to privately identify and answer highly correlated low-dimensional marginals. Their main differences lie in their methodology for selecting these low-dimensional marginals and in how they represent the data distribution from these noisy marginals. For example, MST synthesizes data using probabilistic graphical models (PGMs) [41], PrivBayes uses a Bayesian model, and PrivSyn iteratively updates an initialized dataset using an algorithm they call GUM.

After these NIST challenges, more advanced statistical methods were proposed. Zhang et al. proposed PrivSyn [71] by organizing and refining DPSyn. Cai et al. introduced PrivMRF [7], and McKenna et al. proposed AIM [40], both of which dynamically select low-dimensional marginals and employ PGMs for synthesis. Moreover, methods such as FEM [65] and RAP/RAP++ [3, 64] treat synthesis as an optimization problem, utilizing FTPL [29, 58, 59] and relaxed projection [45], respectively, to refine initialized datasets using adaptively selected marginals. More recently, Liu et al. [36] proposed Private-GSD, which can apply genetic algorithms to adjust datasets based on any selected marginals iteratively.

Deep learning Methods. In addition to statistical methods, deep learning models have been widely explored for DP tabular data synthesis. In this work, we use the category "deep learning model" to broadly encompass all neural network-related approaches, even though some of them do not have a "deep" network structure. In NIST 2018, there was an effort to utilize GANs to generate data. However, this method, called DP-GAN [66], did not demonstrate good performance. Further attempts on generative adversarial networks (GANs), such as DP-GAN [66], DP-WGAN [55], DP-CGAN [61], PATE-GAN [28], and DP-CTGAN [18], also demonstrated limited performance before. Therefore, we omit the comparison of this class of methods.

More recent deep learning approaches, including GEM [37] and DP-MERF [23], represent generative network-based advancements. GEM combines generative networks with adaptive marginal selection mechanisms, while DP-MERF employs random Fourier feature loss to train generative

networks. Besides, TabDDPM [31] leverages diffusion models' representational power to fit target data directly. While TabDDPM was not originally designed for DP, it achieves state-of-the-art performance among non-DP methods and can be adapted for DP synthesis using DP-SGD [16]. Consequently, we include TabDDPM in our analysis.

Our work focuses on recently proposed methods that have not been well compared previously and those representing the current state-of-the-art, as summarized in Table 1. Here, we must emphasize that specifically, we don't consider some LLM-based methods [52, 62]. That is because LLMs are trained on extensive public datasets, which will introduce evaluation bias and affect our further comparison of algorithm modules.

3.2 Existing Benchmark Works

In addition to the proposed algorithms, there are several benchmark studies [16, 17, 26, 60, 67] that investigate the problem of DP data synthesis. However, previous works all have some weaknesses, summarized as follows.

- **Lack of unified comparison setting**. Du et al. [16] have made an experimental evaluation of current works, but ignore the importance of a unified setting (e.g., preprocessing), which may significantly influence the comparison fairness.
- **Not include recent advanced works**. Du et al. [16] and Tao et al. [60] focus on utility evaluation, but they both lack investigation for some advanced methods, such as RAP++ and Private-GSD. Fan et al.'s work [17] only focuses on GAN-based methods.
- **Lack of deep analysis**. Yang et al. [67] provide a survey of many methods and further investigate distributed data synthesis. However, they do not delve deep into these algorithms' working principles. Du et al.'s work and Tao et al.'s work also have a similar weakness in lacking deep algorithm analysis.
- **Lack of comprehensive experiments**. Hu et al. [26] provide analysis for a wide range of DP synthesis algorithms, not only about tabular data synthesis but also trajectory data and graph data. This work, though trying to make an in-depth analysis, lacks comprehensive experiments.

Except for these works, there are other "benchmark-like" works on different aspects of this problem. For instance, Ganev et al. [20] discuss the importance of discretization in tabular data synthesis. Moreover, Stadler et al. [56] focus more on the quantitative evaluation of privacy gain. These works provide in-depth research from different perspectives but lack a more comprehensive viewpoint.

Realizing the weaknesses of current research, our work aims to address these issues by proposing a standardized algorithmic framework (Section 4), providing rigorous analysis (Section 5, Section 6, and Section 7), and conducting detailed experiments (Section 9).

4 Framework Overview

Some previous works [26, 37, 39] have identified that current methods have several common working patterns and proposed unifying algorithmic frameworks, focusing primarily on feature selection and dataset synthesis. However, when dealing with a complex dataset, preprocessing it into a more manageable form should also be a necessary part of the algorithm. As presented Figure 1, our framework extends these approaches by incorporating a data preprocessing module. This section gives an overview of our framework and presents how modules work together to form a complete and cohesive data synthesis pipeline.

Preprocessing. As observed in our evaluation, many datasets contain attributes with large domain sizes (e.g., exceeding 10^5 in Loan dataset [2]), which pose significant challenges for data synthesis. For statistical methods, large attribute domains lead to expansive, low-dimensional marginals,

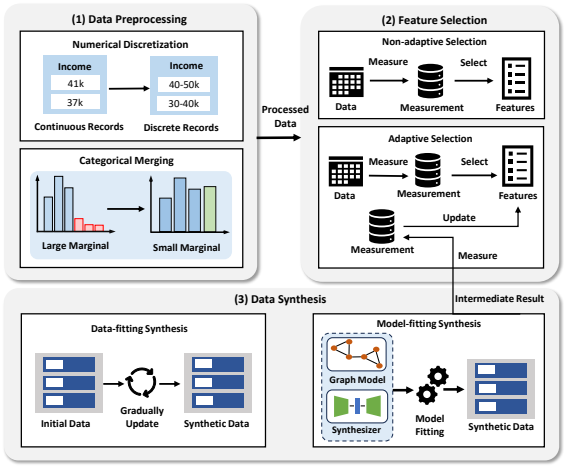


Fig. 1. The proposed unified framework. The dataset is first preprocessed and then represented by some selected features. Finally, using the selected features, the synthesis algorithm generates data as the output of the workflow.

which will introduce excessive DP noise during synthesis and degrade performance. For deep learning methods, a large domain requires a synthesizer with a substantial model size to learn the relationship between different attributes, leading to slower training and higher resource demands. Additionally, larger model sizes require a higher scale of DP noise under the same privacy budget compared to smaller synthesizers, degrading synthetic performance [22].

Therefore, preprocessing is important and non-negligible in algorithm workflows, which can reduce the dimensionality of marginals by effectively merging those with similar characteristics and compressing the domains of the attributes. However, as shown in Table 1, preprocessing is often overlooked by prior works [3, 7, 23, 31, 36, 37, 40]. This omission in the algorithm pipeline may affect the fairness of comparisons between methods.

Feature Selection. Even after effective preprocessing, the domain of the whole dataset increases exponentially with the number of attributes, making methods that rely on a histogram representation computationally infeasible. A practical approach is to utilize some representative local data features, such as *low-dimensional marginals*, to approximate the full joint data distribution [7, 36, 40, 71]. Therefore, the second step in the framework is to measure the representativeness of the features and select necessary features. To further analyze how to better select features in Section 6, we will categorize existing strategies into two categories:

- **Non-adaptive Feature Selection**. A straightforward approach to this step involves performing one-shot feature selection. Some methods, such as DP-MERF and Private-GSD, predefine a fixed set of features without selection. While others, like PrivSyn, measure and select features in one step.
- **Adaptive Feature Selection**. Adaptive methods iteratively obtain new features based on intermediate feedback from previous selection steps to optimize algorithms. For example, AIM selects one new marginal at each iteration step and fits the model based on all selected marginals, which then serves as the reference for the next round of marginal selection. Moreover, DP-SGD iteratively calculates DP gradient features to update the model.

Data Synthesis. The third step in the framework is to synthesize data that aligns well with the features selected in the feature selection step. Current methods apply a wide range of algorithms

Algorithm 1: DP Rare Category Merge

Input: dataset D, merge threshold parameter θ, unique value threshold β, DP parameter ρ_2
Output: preprocessed dataset D

$V_c \leftarrow$ categorical variables with domain size $\geq \beta$;
$\rho' \leftarrow \rho_2 / |V_c|$;
$\sigma = \sqrt{1/(2\rho')}$;
for $j \in V_c$ **do**
  $b \leftarrow$ 1-way marginal of attribute A_j;
  $\hat{b} = b + \mathcal{N}\left(0, \sigma^2\right)$;
  for $i = 1 : |\hat{b}|$ **do**
    $\theta' \leftarrow \max\left\{\theta \cdot \sum \hat{b},\ 3\sigma\right\}$; ▷ Merging threshold
    if $\hat{b}[i] < \theta'$ **then**
      replace i-th value with the rare encoding value;
    end
  end
end
return D

to achieve this synthesis step. Broadly, there are two approaches: fitting the dataset or fitting the model.

PrivSyn manually adjusts data records to match the marginals, while Private-GSD achieves this by genetic algorithm [57]. RAP and RAP++ use the relaxed projection mechanism to optimize the records of an initialized dataset. These methods are constructed by adjusting records. Some other methods fit models for data generation. PGM [41] is a classical graphical model for tabular data synthesis, which utilizes a tree-like model to represent the distribution of data. deep learning models such as generative networks and diffusion models have also been applied here.

It is also notable that this division is not mutually exclusive. We can regard an initialized dataset as a model where each data record is a model parameter, so adjusting data can also be regarded as fitting a model. Our characterization is primarily for explaining algorithm intuition.

5 Preprocessing

As mentioned in Section 3, when synthesizing complex datasets, we may encounter attributes with high cardinality. For example, an address or income attribute can have thousands of distinct values. Conducting statistics on such attributes is infeasible due to the high memory requirements and long execution times. Therefore, a preprocessing step is necessary for algorithm comparison. We surveyed some previous works [20, 39, 71] and summarized the data types requiring preprocessing into two types: categorical and numerical. Before introducing this section, we need to clarify a basic assumption: domain information is considered public knowledge. This assumption is reasonable in many cases. For example, the domains of personal attributes in census data are well documented on the IPUMS website [27].

5.1 Categorical Attributes Preprocessing

Some prior works [39, 71] have proposed a 3σ merging strategy to preprocess categorical variables, where categories with counts below 3σ (where σ represents the DP noise standard deviation in the counting process) are combined.

Algorithm 2: PrivTree Binning

Input: dataset D, unique value threshold β, divide parameter θ, privacy parameter ρ_1
Output: preprocessed dataset D

Set $T = \emptyset$
$V_n \leftarrow$ numerical variables with domain size $\geq \beta$
$\beta_0 \leftarrow 2$
$\lambda' \leftarrow \frac{2\beta_0 - 1}{\beta_0 - 1} \cdot \sqrt{\frac{|V_n|}{2\rho_1}}$
$\delta' \leftarrow \lambda' \cdot \ln \beta_0$
for $j \in V_n$ **do**
    $\mathcal{T} \leftarrow \text{PrivTree}(D[j], \lambda', \delta', \theta)$
    Append $\mathcal{T}$ to T
end
Apply T to discretize dataset D
return D

By controlling each category's frequency to be large enough (larger than 3σ), this approach helps mitigate the impact of noise. However, this method has limitations: when we have a large privacy budget, 3σ could be a small value. If we continue using 3σ as the threshold for combining, we may be able to achieve high accuracy, but we cannot reduce the attributes' domain size to ensure algorithm efficiency.

In response to these limitations, we improve the 3σ merging method, as shown in Algorithm 1, by applying a dual merging threshold $\max\{3\sigma,\ \hat{n}\theta\}$. Here $\hat{n}$ is the privately measured number of records in the dataset and θ is the threshold parameter. By introducing a fixed threshold, we can avoid the case when σ is too small to reduce the attribute's domain complexity.

5.2 Numerical Attributes Preprocessing

Continuous numerical variables often exhibit dense distributions within specific intervals, with numerous unique values. Different from categorical attributes, these unique values have numerical correlation, making value combining (like what we do for categorical attributes) impossible. Thus, discretization is a proper preprocessing method for numerical attributes. Here, we outline two potential discretization approaches:

Uniform Binning. Some previous works [13, 20] use uniform binning for continuous data preprocessing. It partitions an attribute's domain into equal-length intervals, relying only on the attribute's domain range and a predefined number of bins. Formally, this method can be expressed as $\text{Uniform Bin}(x) = \left\lfloor \frac{x - x_\ell}{h} \right\rfloor$, where x_ℓ is the lower bound of the attribute's domain, and h is the length of the uniform interval determined by the bin number.

Uniform binning is advantageous because it requires no detailed data information, avoiding the need for additional privacy budget allocation. However, it has significant drawbacks, particularly for attributes with uneven distributions. For example, when data is highly concentrated around specific values, uniform binning can lead to inefficient binning, as it may allocate unnecessary bins to sparsely populated areas.

PrivTree. A limitation of uniform binning is its reliance on a predetermined number of bins, which introduces concerns about hyperparameter selection. To address this, PrivTree decomposition [60, 70] can be used as a self-adaptive discretization method.

PrivTree employs a tree structure to iteratively divide the domain of an attribute, with splits continuing until intervals contain only a small number of records. However, since PrivTree utilizes sensitive data for domain division, it requires a fraction of the privacy budget to guarantee differential privacy. We use PrivTree on multiple attributes whose domain size is larger than a threshold, with algorithm details in Algorithm 2. We provide the proof of DP guarantee of this algorithm in the appendix of our full paper [10].

6 Feature Selection

Selecting features determines the performance of many algorithms. Similar to preprocessing methods, it is important for statistical methods, while some deep learning methods can automatically learn the characteristics of the dataset by training models. In this section, we will briefly introduce the currently proposed selection methods and analyze them.

6.1 Existing Selection Methods

Non-adaptive Feature Selection. Non-adaptive methods perform all feature computations and operations at the beginning of the algorithm based on the characteristics of the marginals. The selected features will then be delivered to the synthesis modules without any further refinement.

Some algorithms directly predefine a set of features instead of paying attention to selecting representative features. DP-MERF [23] leverages random Fourier features to capture correlations among numerical variables, while employing 2-way marginals for categorical variables. In addition, some methods with strong fitting ability can work on all two-way marginals. For example, Liu et al. [36] conduct experiments on all two-way marginals to demonstrate the performance of Private-GSD.

Instead of predefining some features, PrivSyn [71] selects the most highly correlated marginals while respecting the privacy budget in one round. They measure each marginal using metric $\mathsf{InDif}_{i,j} = \left|M_{i,j} - M_i \times M_j\right|$, where M denotes the attribute marginal. The selection process involves minimizing the expected error

$$\sum_i \left(N_i x_i + \mathsf{InDif}_i(1 - x_i)\right),$$

where $x_i \in \{0, 1\}$ denotes the selection decision and N_i is DP noise.

Adaptive Feature Selection. Different from non-adaptive methods, adaptive methods continuously update feature selection with feedback from the previous selection steps. In each synthesis round, adaptive feature selection conducts many queries on the current estimation, and the features with large errors are selected. These features will be used for the selection of future rounds. This strategy is used by methods like RAP [3], RAP++ [64], GEM [37], PrivMRF [7] and AIM [40]. These methods differ in several key aspects:

- First, in terms of initialization, PrivMRF uses a carefully designed criterion to select a small set of features as the starting point, while AIM initializes with all 1-way marginals. In contrast, RAP, RAP++ and GEM do not specify any initialization.
- Secondly, the selection criteria also differ among these methods. Both AIM and PrivMRF's marginal selection criteria include a punishment term proportional to the marginal scale, allowing for a balance between noise level and feature representativeness. Other methods measure the features without a penalty term, which is one-sided and potentially introduces more errors.
- Finally, the selection mechanism varies: AIM and GEM utilize the exponential mechanism [8], whereas RAP and RAP++ employ the Gumbel mechanism [3], allowing them to select more than one feature in each round; PrivMRF takes a different approach, directly adding Gaussian noise to the selection criteria for selection and select the largest one.

6.2 Analysis for Selection Algorithms

Some previous studies [7, 37, 40] have discussed the important properties of a good selection mechanism. In this subsection, we formally investigate some aspects of it. Since all methods, except for DP-MERF, which predefines features without selection, focus on marginal selection, we only discuss marginal selection here.

Importance of Scale Penalty Term. For any marginal M, we have two choices: choose and privatize it or do not choose it. Let the marginal estimated by available information be M_0, and the privatized marginal be $\hat{M}$. We have $\hat{M} = M + \mathcal{N}(0, \sigma)$, where σ is known privacy budget. The expected error can be derived as,

$$\begin{cases} \text{Selection error: } \|M - \hat{M}\|_1 = n\sigma\sqrt{2/\pi} \\ \text{Unselection error: } \|M - M_0\|_1. \end{cases}$$

Therefore, we can compare these two errors, denoted as $\|M - M_0\|_1 - n\sigma\sqrt{\frac{2}{\pi}}$, to determine the selection result, which demonstrates the importance of the scale penalty term. This equation is also how some selection criteria involve the scale penalty term.

Superiority of Adaptive Selection. Establishing the superiority of the adaptive selection strategy is challenging in the absence of prior information about the generation module. Our analysis, therefore, proceeds under a set of reasonable assumptions. First, we consider the case of selecting 2-way marginals. Second, we assume that conditional independence is used for marginal selection. Specifically, assuming that before selecting (A_i, A_j), we have already fitted marginals $(A_i, A_1, \cdots, A_k)$ and $(A_j, A_1, \cdots, A_k)$, so that we can use them to estimate the distribution of (A_i, A_j) as

$$\begin{aligned} \hat{\Pr}[A_i, A_j] = \sum \Pr[A_1, \cdots, A_k] \cdot \\ \Pr[A_i | A_1, \cdots, A_k] \Pr[A_j | A_1, \cdots, A_k]. \end{aligned} \tag{1}$$

Before selecting the marginal (A_i, A_j), the non-adaptive methods do not have any intermediate results and can only use independent 1-way marginals to measure its representativeness. This independent measurement can be written as $\mathbb{D}_{\mathrm{KL}}\left(\Pr[A_i, A_j] \,\middle\|\, \Pr[A_i]\Pr[A_j]\right)$, where $\mathbb{D}_{\mathrm{KL}}$ is the KL Divergence [32]. The adaptive methods, because they select features based on intermediate synthesis results, will have a conditional estimation as $\mathbb{D}_{\mathrm{KL}}\left(\Pr[A_i, A_j] \,\middle\|\, \hat{\Pr}[A_i, A_j]\right)$, which is the true KL divergence when choosing marginal (A_i, A_j). Here $\hat{\Pr}[A_i, A_j]$ is defined in Equation (1). Now we give the following theorem to prove the superiority of adaptive selection.

Theorem 2. *For any pair of attributes (A_i, A_j), the KL divergence error of conditional estimation is always no larger than that of independent estimation. Formally, we have*

$$\mathbb{D}_{KL}\left(\Pr[A_i, A_j] \,\middle\|\, \hat{\Pr}[A_i, A_j]\right) \leq \mathbb{D}_{KL}\left(\Pr[A_i, A_j] \,\middle\|\, \Pr[A_i]\Pr[A_j]\right)$$

Here $\hat{\Pr}[A_i, A_j]$ is defined in Equation (1).

The proof of Theorem 2 is deferred to the appendix of our full paper [10]. In essence, this theorem demonstrates that under our assumptions, non-adaptive methods tend to overestimate the representativeness of features under KL divergence, whereas adaptive methods can correct this error by leveraging intermediate results. The more general comparisons between adaptive and non-adaptive selection strategies will be conducted in our experiments. Another notable fact is that compared to the non-adaptive methods, the adaptive methods require multiple rounds of data synthesis or computation during the feature selection, which can lead to higher time consumption as a trade-off for their superiority in utility.

7 Data Synthesis Module Comparison

Previous sections have shown that current solutions utilize various techniques to generate data on selected features. This raises critical questions about their utility and efficiency. Thus, we will analyze this problem in this section. For data-fitting methods, there are GUM, Genetic algorithm, and Relaxed Projection. For the model-fitting type, we consider PGM and (deep) generative network.

GUM. GUM, used by PrivSyn, is an iterative adjustment method that modifies values in the initial dataset to align with the selected marginals. We assume that GUM merges marginals to k cliques of sizes $\{c_1, \cdots, c_k\}$. Since the maximum number of operations to fit each value in the marginals is no more than the size of the synthetic dataset, the time complexity of GUM is $O\left(\sum_{i=1}^{k} Tc_i n\right)$, where T is the number of update iterations, n represents the synthetic dataset size, respectively. Because GUM strictly controls the clique size c_i, ensuring we have small cliques and simplifying the update process (e.g., we can choose a small T to reach convergence). This guarantees the efficiency of GUM. However, it fits marginals one by one, overlooking overall correlations, limiting its utility.

Genetic Algorithm. Genetic algorithms [57] use mutation and crossover operations to adjust datasets. The complexity depends on the number of mutations and crossovers per iteration. For instance, the algorithm by Liu et al. [36] has a complexity of $O\left(T(P_m + P_c)\right)$, where T is the number of iterations, and P_m and P_c represent mutations and crossovers per iteration, respectively. The execution time of genetic algorithms is strongly influenced by the number of tuning rounds, which often exceeds the dataset size when handling complex datasets with diverse values.

Relaxed Projection. Relaxed projection mechanism [3, 64] treats the dataset as a trainable model, optimizing it to match marginals. Given T optimization rounds, synthetic data size n, and data dimension d, the time complexity is $O(Tnd)$. This method can be regarded as both a model-fitting and data-adjusting approach. The complexity is driven by the size of the synthetic data and the number of optimization rounds. High-dimensional datasets with large attribute domains can result in an inflated encoded data dimension, making optimization difficult to converge and more time-intensive.

PGM. PGM [41] constructs a junction tree of marginal cliques $C_1, C_2, \cdots, C_k$, ensuring that the intersection set S_i of any two cliques appears only in those marginals. The overall distribution is approximated as:

$$\Pr[A_1, \cdots, A_d] \approx \Pr[C_1] \cdot \prod_{i=2}^{k} \Pr[C_i \setminus S_i \mid S_i]. \tag{2}$$

Let T be the number of training iterations, and let k cliques have sizes $\{c_1, \cdots, c_k\}$. The total complexity is $O\left(\sum_{i=1}^{k} Tc_i + nk\right)$, which includes model training and data genration complexity. PGM's efficiency depends on constructing reasonable cliques, which are automatically determined by the junction tree. Densely selected marginals can lead to large cliques, greatly increasing time costs.

Generative Network. Generative networks [37] rely on parameters m, batch size b, and training iterations T. The training complexity is $O(Tmb)$. Generating n synthetic records results in a total complexity of $O(Tmb + mn)$. The efficiency is affected by network design and training hyperparameters, which can be challenging for high-dimensional datasets.

In summary, we find that model-fitting type methods heavily depend on the construction of the generative model. While this approach can achieve high efficiency, it also risks encountering the curse of dimensionality. In contrast, data-fitting methods are often limited by the complexity of the data itself, which can significantly impact algorithm efficiency.

Table 2. Summary of investigated datasets. We report the number of records, the number of attributes, numerical attributes, categorical attributes, and minimum and maximum attribute domain size.

Name	#Records	#Attr	#Num	#Cat	Min/Max Domain
ACSincome (INC) [14]	55320	10	2	8	2~93
ACSemploy (EMP) [14]	37881	17	1	16	2~92
Bank (BK) [44]	45211	16	6	10	2~6024
Higgs-small (HIG) [1]	98049	28	28	1	2~73715
Loan (LN) [2]	134658	42	25	17	2~93995

8 Experimental Setup

Our evaluations aim to (1) compare all well-established methods within our unified framework; (2) explore and verify the importance of preprocessing in DP tabular synthesis tasks; (3) investigate feature selection and synthesis modules of these methods for a more fine-grained comparison. The comparison results can validate previous theoretical analysis and guide method selection for different modules within the framework. To achieve these goals, we design and conduct three main experiments. We summarize them and some important findings as follows.

- *Overall Evaluation*: We evaluate method performance across metrics and datasets. A key observation is that statistical methods like AIM and PrivMRF demonstrate better synthesis utility, while under current implementations, deep learning methods can achieve higher time efficiency.
- *Preprocessing Investigation*: These experiments focus on the preprocessing investigation by comparing preprocessed datasets with raw ones. Through experiments, we prove that preprocessing is essential in reducing algorithm complexity without introducing much synthesis error.
- *Module Comparison*: Finally, we evaluate the effectiveness of different feature selection and synthesis modules by fixing one module and reconstructing algorithms by changing the other module. We find that (1) the adaptive selection strategy can help the algorithm achieve a higher synthesis utility; (2) existing different synthesis modules show their limitations in either utility or scalability.

8.1 Datasets

To make a comprehensive comparison, we would expect datasets to (1) vary in record and domain size, and attribute distribution to better show method performance on datasets of various complexity, and (2) differ in the proportion of numerical and categorical attributes to assess the preprocessing methods. Therefore, we choose five datasets used in previous work [31, 36, 40, 71] as our datasets, which are ACSincome, ACSemploy, Bank, Higgs-small and Loan. The brief information about these datasets is provided in Table 2.

Furthermore, in Figure 2, we plot the heat maps on absolute values of pairwise correlations across different datasets. Darker colors in the heat map mean higher correlation. We also plot the histograms of the attribute distribution's Shannon entropy, defined as $H(X) = -\sum_{i=1}^{k} p_i \ln p_i$. The larger the entropy, the more informative and complex the distribution is. From Figure 2, we can observe that ACSincome and ACSemploy have stronger correlations between attributes, while attribute distributions in the other three datasets have higher entropy, indicating greater complexity.

8.2 Implementations

In our experiments, we consider PrivSyn , PrivMRF , RAP++ , AIM , Private-GSD , GEM , DP-MERF , and TabDDPM . We do not include RAP in the experiments because RAP++ directly improves upon it. Moreover, notice that Private-GSD is a synthesis algorithm; we use its one-shot 2-way marginal

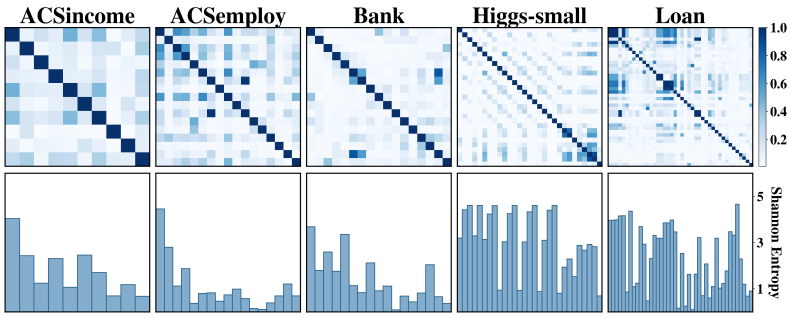


Fig. 2. Heat map of pairwise absolute correlations and histogram of attribute distribution's Shannon entropy.

version in the overall evaluation, which is also used in their original work, and pay more attention to its performance in module comparison. Finally, we train TabDDPM under DP-GSD by *Opacus* [68]. We repeat all evaluations five times and report the average results. The DP parameter δ is set to be 10^{-5} by. We, by default, employ uniform binning and rare category merging preprocessing methods. The detailed hyperparameter settings of the studied methods, including preprocessing algorithms, are provided in the appendix of our full paper [10].

We keep our implementation of these methods aligned with their released versions. PrivSyn and AIM are executed on CPU, while the other methods are executed with GPU. This is because PrivSyn and AIM are designed for CPU and executing them on GPU may not be able to bring better performance. For instance, PrivSyn adjusts data in cliques one by one, which does not require high-dimensional computation. We also try AIM (PGM) on GPU (also with the officially released code from its authors) and compare the results with those from CPU. A brief result is shown in Table 3. We can observe that there is no significant improvement in algorithm performance after executing on GPU.

8.3 Evaluation Metrics

Various evaluation approaches have been proposed [16, 40, 71]. However, most current works focus on comparing synthesis utility, or how similar synthetic data is to real data. While utility is important, algorithm efficiency is also a critical factor in algorithm selection. Therefore, we mainly consider the following metrics.

Machine learning efficacy (higher is better). A widely accepted metric for evaluating generated data is training downstream machine learning (ML) models on the generated data and assessing their performance on test data. The previous works [16, 31, 71] typically select one or more ML models as downstream tasks. It's important to note that a large number of models do not necessarily lead to a fairer evaluation. Generally, simpler models have weaker data-fitting capabilities, which cannot reach fair conclusions. In our comparison, we therefore selected four deep learning models known for strong performance across various datasets: MLP, CatBoost, XGBoost, and Random Forest. We report the average F1 score on held-out test data as our metric value, and other related metrics, such as AUC and accuracy, are provided in the appendix of our full paper [10].

Another justification is that we use the test dataset instead of the training dataset for evaluation for this and the remaining metrics. While both approaches have been employed in previous works, we opt to use test data for evaluation due to the belief that it better reflects an algorithm's generalization ability.

Query Error (lower is better). Making queries [9] is a commonly used data analysis technique, which can also be conducted to measure relatively high-dimensional similarity due to its high efficiency. Here, we consider using the 3-way marginal query method employed by Du et al. and

Table 3. Results of AIM (PGM) on different hardware (CPU and GPU). These results are obtained on ACSIncome dataset.

ε	Method	MLE↑	Query Err.↓	Fid Err.↓	Time
0.2	AIM (CPU)	0.77	0.0016	0.093	2.0 min
	AIM (GPU)	0.75	0.0024	0.105	2.5 min
1.0	AIM (CPU)	0.78	0.0011	0.064	6.3 min
	AIM (GPU)	0.78	0.0011	0.064	37.0 min

McKenna et al. [16, 41], which utilizes the statistical ℓ_1 error of frequency query results to reflect the magnitude of the error. Formally, the query error can be expressed as, $\mathbb{E}_{r\in R}\left|q_r(D_{syn}) - q_r(D_{test})\right|$, where q_r refers to the query function, which is a combination of range query (for numerical attributes) and point query (for categorical attributes). $\mathbb{E}$ is the mathematical expectation, and R refers to the set of all 3-way marginals. Moreover, due to complex real applications on synthetic data, we also worry that simple frequency queries, even though fundamental, may be insufficient to cover them. Therefore, we also consider **Conditional Query Error** to simulate `Join` and `GroupBy` in SQL (fix one attribute as the key column and query the frequency of the other attribute). This metric serves as a supplementary evaluation reference, providing comprehensive comparisons.

Fidelity Error (lower is better). Marginal Fidelity is precise in evaluating low-dimensional similarity, such as average 2-way marginal discrepancy. Total variation distance (TVD) can be used for such measurement, which has also been utilized in some studies [60, 62]. We define the TVD as $\frac{1}{2}\sum_{1\le i\le j\le d}\left|M^{\text{syn}}_{i,j} - M^{\text{test}}_{i,j}\right|$, where $M^{\text{syn}}_{i,j}$ and $M^{\text{test}}_{i,j}$ are the real 2-way marginals determined by the synthetic dataset and test dataset, respectively.

Running Time (lower is better). A straightforward measurement of algorithm efficiency is the execution time when generating the same amount of data. The running time does not include the preprocessing step and only counts the time spent in feature selection and data synthesis modules.

In addition to our primary evaluation criteria, there are many other widely applied metrics for tabular data, such as Wasserstein distance [49] and Cramer's V measure [11]. We do not include these metrics in our evaluation mainly because of their feasibility in complex cases and overlap with existing metrics. For example, Wasserstein distance can be infeasible in terms of computation time when dealing with high-dimensional and complex data. Moreover, Cramer's V measure and other correlation measurements cannot provide deterministic information about data distributions. In other words, even if two distributions are different, they can have similar correlation results. Furthermore, correlation information can be covered by fidelity error, which is based on a precise measurement of total distribution. We will further validate this in the appendix of our full paper [10].

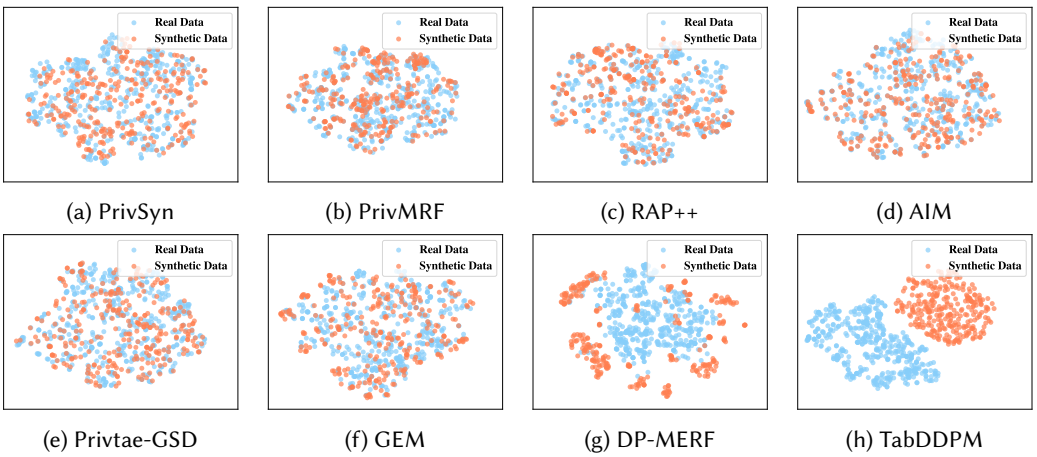
9 Experimental Results

9.1 Overall Evaluation

Utility Comparison. We provide detailed results on machine learning efficacy, query error, and fidelity error in Table 4, and show some brief results of conditional query error in Table 5. To make the result more straightforward, we utilize the t-SNE [63], a dimensionality reduction technique, to visualize the synthesis results of all methods on the Bank dataset in Figure 3. By reducing the dimensionality of the dataset with t-SNE technique and plotting the scatter distribution, these visualizations reveal the distribution overlap between synthetic and real data, demonstrating synthesizers' ability to generate data resembling the original.

Table 4. Overall utility of synthetic data under different methods. The results with the best performance are highlighted in bold (due to the limited number of digits displayed, some data may appear equal in the table, but there is actually an order in their values). Ground Truth is obtained by comparing real data with test data.

Dataset	**ACSincome**			**ACSemploy**			**Bank**			**Higgs-small**			**Loan**		
ML Efficacy ↑	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$
PrivSyn	0.43	0.39	0.42	0.40	0.43	0.39	0.47	0.47	0.47	0.40	0.43	0.43	0.25	0.26	0.26
PrivMRF	0.73	0.78	0.78	0.72	0.80	0.81	0.62	0.69	**0.71**	0.50	0.64	0.64	**0.52**	**0.52**	**0.52**
RAP++	0.66	0.73	0.77	0.74	0.77	0.80	0.65	0.69	0.67	0.52	0.53	0.54	0.45	0.42	0.43
AIM	**0.76**	**0.78**	**0.78**	**0.78**	**0.80**	**0.81**	**0.67**	**0.71**	0.71	**0.63**	**0.64**	**0.67**	0.52	0.52	0.52
Private-GSD	0.76	0.77	0.77	0.72	0.73	0.72	0.47	0.48	0.49	0.48	0.47	0.49	0.25	0.24	0.25
GEM	0.70	0.68	0.66	0.67	0.70	0.69	0.51	0.56	0.53	0.51	0.52	0.52	0.50	0.49	0.51
DP-MERF	0.65	0.67	0.71	0.58	0.71	0.66	0.60	0.57	0.55	0.53	0.56	0.57	0.32	0.18	0.18
TabDDPM	0.41	0.41	0.39	0.48	0.42	0.51	0.47	0.47	0.47	0.36	0.35	0.34	0.24	0.24	0.24
Ground Truth	0.79	0.79	0.79	0.81	0.81	0.81	0.76	0.76	0.76	0.72	0.72	0.72	0.54	0.54	0.54
Query Error ↓	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$
PrivSyn	0.003	0.002	0.002	0.008	0.004	0.004	0.007	0.004	0.003	0.009	0.004	0.003	0.006	0.005	0.004
PrivMRF	0.002	0.001	0.001	0.004	0.002	0.002	**0.005**	0.003	0.003	0.005	0.005	0.003	**0.005**	**0.005**	**0.004**
RAP++	0.019	0.005	0.003	0.029	0.009	0.003	0.014	0.006	0.005	0.035	0.029	0.028	0.020	0.014	0.011
AIM	**0.002**	**0.001**	**0.001**	**0.004**	**0.002**	**0.001**	0.007	**0.002**	**0.002**	**0.005**	**0.003**	**0.003**	0.005	0.005	0.004
Private-GSD	0.004	0.003	0.002	0.026	0.026	0.026	0.044	0.044	0.043	0.044	0.044	0.044	0.038	0.037	0.036
GEM	0.014	0.017	0.016	0.010	0.006	0.006	0.118	0.021	0.022	0.066	0.065	0.065	0.030	0.030	0.029
DP-MERF	0.019	0.018	0.024	0.039	0.037	0.036	0.038	0.035	0.036	0.039	0.035	0.034	0.006	0.006	0.006
TabDDPM	0.066	0.064	0.060	0.088	0.067	0.079	0.074	0.071	0.088	0.106	0.106	0.107	0.067	0.066	0.070
Ground Truth	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Fidelity Error ↓	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$	$\varepsilon = 0.2$	$\varepsilon = 1$	$\varepsilon = 5$
PrivSyn	0.15	0.12	0.12	0.12	0.09	0.09	0.24	0.13	0.12	0.29	0.20	0.15	0.34	0.35	0.36
PrivMRF	0.11	0.07	0.05	0.07	0.04	0.03	0.13	**0.07**	**0.06**	0.21	**0.16**	0.16	**0.31**	**0.24**	**0.23**
RAP++	0.52	0.24	0.19	0.30	0.13	0.07	0.43	0.36	0.36	0.69	0.65	0.65	0.66	0.58	0.55
AIM	**0.09**	**0.06**	**0.05**	**0.05**	**0.03**	**0.02**	**0.11**	0.09	0.09	**0.19**	0.17	**0.14**	0.35	0.32	0.29
Private-GSD	0.23	0.21	0.20	0.22	0.22	0.22	0.52	0.52	0.52	0.65	0.65	0.65	0.67	0.66	0.66
GEM	0.26	0.28	0.27	0.15	0.08	0.09	0.76	0.21	0.23	0.57	0.57	0.36	0.53	0.52	0.52
DP-MERF	0.52	0.51	0.50	0.34	0.34	0.32	0.47	0.48	0.48	0.60	0.56	0.54	0.91	0.92	0.93
TabDDPM	0.78	0.77	0.71	0.60	0.51	0.57	0.79	0.78	0.82	0.70	0.81	0.88	0.95	0.95	0.94
Ground Truth	0.05	0.05	0.05	0.02	0.02	0.02	0.03	0.03	0.03	0.12	0.12	0.12	0.10	0.10	0.10

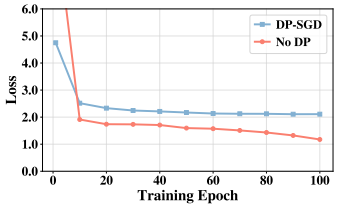


(a) PrivSyn (b) PrivMRF (c) RAP++ (d) AIM

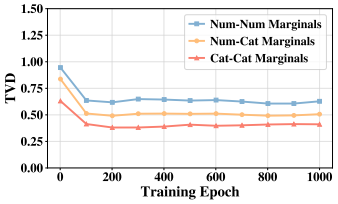
(e) Privtae-GSD (f) GEM (g) DP-MERF (h) TabDDPM

Fig. 3. T-SNE scatter plots of synthesis results on Bank dataset under $\varepsilon = 1.0$

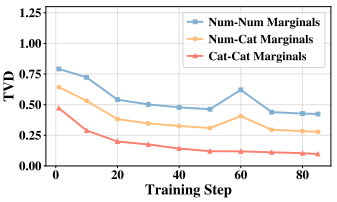
The first clear conclusion is that PrivMRF and AIM achieve the best utility metrics among all methods. Another straightforward finding is that statistical methods generally perform better than deep learning approaches. Both AIM and PrivMRF rely on graphical models, and the quantitative



(a) Training loss of TabDDPM with and without DP on Bank dataset under $\varepsilon = 1.0$.



(b) TVD of marginals between generated data and real data when training DP-MERF.



(c) TVD of marginals between generated data and real data when training GEM.

Fig. 4. Figures for analyzing different methods in Section 9.1

Table 5. Conditional query errors of different methods under $\varepsilon = 1.0$.

Method	ACSincome	ACSemploy	Bank	Higgs-small	Loan
Privsyn	0.019	0.035	0.033	0.029	0.058
PrivMRF	**0.012**	0.013	0.026	0.026	**0.053**
RAP++	0.021	0.026	0.064	0.098	0.098
AIM	0.013	**0.011**	**0.022**	**0.025**	0.057
Private-GSD	0.016	0.014	0.039	0.110	0.093
GEM	0.026	0.028	0.053	0.069	0.077
DP-MERF	0.308	0.314	0.215	0.099	0.455
TabDDPM	0.092	0.072	0.136	0.109	0.143

evaluations of GEM, DP-MERF, and TabDDPM are worse than most other statistical methods in some metrics, such as RAP++ and PrivSyn in fidelity error.

Utility & Efficiency Trade-off. We present the relationship between utility and efficiency on different algorithms in Figure 5. Although AIM and PrivMRF exhibit excellent utility performance, they require more execution time, representing a trade-off in utility. PrivMRF is slightly more efficient than AIM because it involves an initialization step, reducing the number of selection rounds. Additionally, all deep learning-based methods, despite having lower generation utility, demonstrate high efficiency. This is partly because some statistical methods run on CPU and also because they involve large-scale computations.

TabDDPM has a weak performance in synthesis utility. Among all the methods, TabDDPM performs poorly in all three dimensions, and in Figure 3, the visualization of the synthesis result does not match the real data distribution. We track the training loss with DP-SGD and compare it with the loss without DP-SGD. In Figure 4a, we can observe that the loss under DP-SGD does not converge well when training. This raises our suspicion of DP-SGD's efficiency. However, the diffusion model still demonstrates excellent performance when generating images [15, 34] under DP-SGD. Therefore, we hypothesize that the failure of TabDDPM is due to its incompatibility with tabular data under a DP-SGD framework. Unlike image data, which is characterized by dense and repetitive features (e.g., visual similarity among images of the same type), tabular data exhibits sparse and highly divergent characteristics (e.g., unique relationships between attributes), which present significant convergence challenges for DP-SGD.

Two GAN-based methods, DP-MERF and GEM, perform differently. Even though GEM and DP-MERF both use deep generative networks for synthesizing, compared to GEM, DP-MERF performs poorly regarding query and fidelity error. One key reason lies in the construction of features. Firstly, random Fourier features embedding, used by DP-MERF, is still approximating the joint distribution of numerical attributes, which inherently introduces approximation error. Furthermore, when dealing with categorical variables, DP-MERF focuses solely on the marginal distributions between categorical variables and the label variable, which overlooks other marginals.

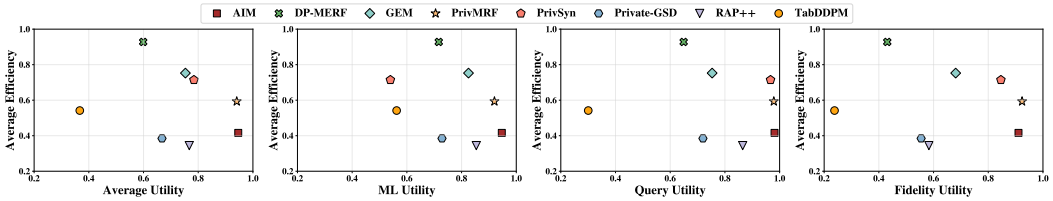


Fig. 5. Scaled utility and efficiency of different algorithms. Average utility is obtained by taking the average of ML efficacy, query error, and fidelity error after normalizing them to $[0, 1]$, guaranteeing their equal contribution to the aggregated metric. Other metrics are directly obtained from the average value of normalized original metrics.

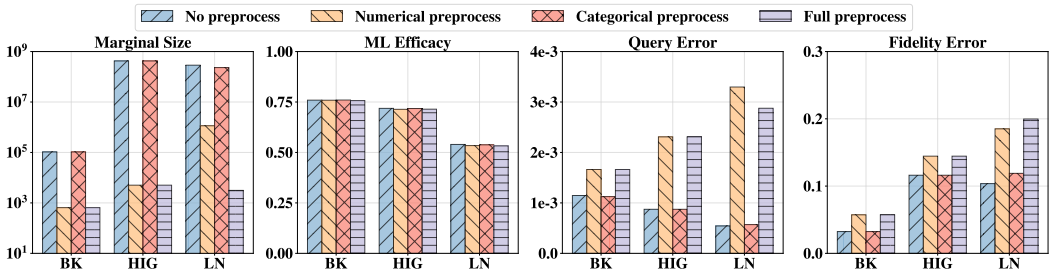


Fig. 6. Metrics under different preprocessing settings. These metrics are obtained by comparing preprocessed raw data with test data. Numerical preprocessing means only conducting numerical discretization, and categorical preprocessing means only conducting categorical rare merging. By default, the results are under the setting that total privacy budget $\varepsilon = 1.0$.

While GEM employs an adaptive marginal selection strategy, continuously adjusting the fitting target.

To verify our hypothesis, we track the fitting errors of different categories of marginals in Figure 4b and Figure 4c. During the DP-MERF training process, the total variation distance (TVD) of most marginals, except those for "categorical-categorical" marginals, does not effectively converge. This, to some extent, supports our thinking. By comparison, GEM achieves better convergence across marginals. Moreover, in Figure 3g, we observe that the data points generated by DP-MERF concentrate in several areas, which we believe is caused by unbalanced feature construction. In contrast, as shown in Figure 3f, the data distribution of GEM is better aligned with the target dataset.

PrivSyn works poorly on machine learning efficacy. Another finding is that PrivSyn performs well on range query error and fidelity error metrics, but underperforms in terms of machine learning efficacy. This scenario aligns with our previous analysis in Section 7, which expresses the concern that GUM focuses more on local marginal cliques and may ignore global relevance. The experiments in Section 9.3 can also support this statement.

9.2 Preprocessing Investigation

In this Section, we will explore the influence of preprocessing on algorithms' performance, and how different preprocessing strategies will influence the quality of synthesis data.

Influence of Preprocessing on Dataset Information. Directly comparing the algorithms' performances with and without preprocessing is difficult because running algorithms on raw datasets is often too time-consuming and computationally complex (e.g., some methods even require more than 24 hours to run on raw datasets). Therefore, we consider comparing the marginal

Table 6. Different algorithms' performance on ACSincome with and without preprocessing under $\varepsilon = 1.0$. Here, "Prep" means preprocessed data, and "Raw" means raw data.

Method	ML Eff. ↑		Query Err. ↓		Fid Err. ↓		Time (min) ↓	
	Prep	Raw	Prep	Raw	Prep	Raw	Preps	Raw
PrivSyn	0.39	0.39	0.002	0.002	0.12	0.12	0.2	0.2
PrivMRF	0.78	0.78	0.001	0.001	0.06	0.07	1.5	0.9
AIM	0.78	0.78	0.001	0.001	0.06	0.06	5.8	6.3
RAP++	0.74	0.74	0.004	0.005	0.22	0.24	13.3	26.8
Private-GSD	0.77	0.77	0.002	0.003	0.20	0.21	9.7	24.9
GEM	0.70	0.68	0.014	0.017	0.25	0.28	0.1	0.1
DP-MERF	0.66	0.67	0.024	0.018	0.52	0.51	0.1	0.1
TabDDPM	0.41	0.40	0.063	0.068	0.76	0.80	6.7	4.1

Table 7. Marginal-based methods' performance on the Higgs-small with different discretization under $\varepsilon = 1.0$. Here, "Tree" means PrivTree and "Uniform" means uniform binning.

Method	ML Eff. ↑		Query Err. ↓		Fid Err. ↓		Time (min) ↓	
	Tree	Uniform	Tree	Uniform	Tree	Uniform	Tree	Uniform
PrivSyn	0.43	0.43	0.005	0.004	0.19	0.20	2.1	6.0
PrivMRF	0.65	0.64	0.005	0.003	0.19	0.16	14.7	7.1
AIM	0.67	0.65	0.005	0.003	0.20	0.19	689.6	18.5
RAP++	0.55	0.53	0.030	0.029	0.55	0.65	74.6	40.0
Private-GSD	0.50	0.47	0.043	0.044	0.57	0.65	58.2	58.3
GEM	0.56	0.54	0.019	0.061	0.29	0.55	0.4	6.1

sizes and utility metrics calculated on preprocessed and raw datasets. The detailed results are shown in Figure 6. A straightforward finding is that preprocessing decreases the complexity of data, with the introduction of only a small error. In Figure 6, the average marginal size significantly decreases after preprocessing (from 10^8 to 10^3 in Higgs-small and Loan datasets). Meanwhile, its negative influences on utility are small enough (change on query error < 0.003, TVD < 0.1). We infer this is because binning can preserve most numerical characteristics, and the low-frequency categorical values only contribute a small proportion of the overall correlation between attributes in the dataset.

Over-preprocessing Problem. In both Algorithm 1 and Algorithm 2, we apply a threshold β to filter out those attributes with small domains to avoid "over-preprocessing". In Table 6, we present the synthesis results for ACSincome with mandatory preprocessing, which does not need preprocessing under threshold β. We can observe that preprocessing does not significantly influence the algorithms' utility. We believe that, because these datasets are inherently simple, preprocessing does not overly distort their information and thus has little impact on the synthesis results. A notable finding is that RAP++ and Private-GSD run faster on the preprocessed dataset. We hypothesize that this is because preprocessing combines similar or less informative values together, which allows these two methods to capture the data characteristics more easily and converge more quickly. In summary, preprocessing can be unnecessary for those simple datasets, but mandatory preprocessing is likely not to be detrimental to synthesis performance.

Numerical Discretization Ablation. Firstly, we summarize the domain sizes under different discretization results in Table 8. It is obvious that PrivTree always generates significantly fewer bins to represent numerical attributes, even for attributes with highly complex value distributions. This approach not only reduces noise in feature measurements but also simplifies the features, raising concerns about losing information. To better demonstrate the influence of discretization, we conduct the ablation study on Higgs-small dataset, which contains the most complex numerical variables among all datasets. For this ablation, we focus on $\varepsilon = 1.0$, as is standard practice [16, 60]. The results are shown in Table 7.

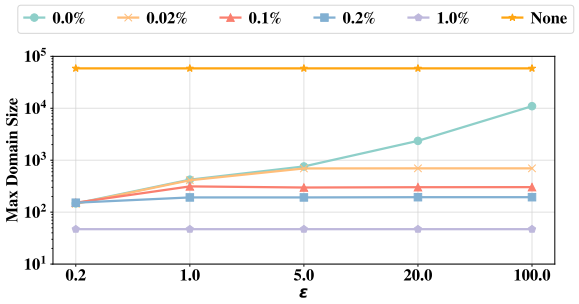


Fig. 7. The maximum attribute domain size of Loan dataset under fixed merging thresholds and different ε. In the figure, "0.0%" means no fixed threshold (merge only by 3σ criteria), and "None" means raw data.

Table 8. Preprocessed numerical attributes' domain sizes under different discretization algorithms. The result is obtained on Higgs-small dataset and under setting $\varepsilon = 1.0$.

Dataset	Min Domain Size			Max Domain Size		
	Raw	PrivTree	Uniform	Raw	PrivTree	Uniform
Bank	505	25	100	6024	28	100
Higgs-small	4870	6	100	60696	18	100
Loan	101	9	100	93995	32	100

In most cases, PrivTree performs better on deep learning tasks but is slightly worse in query errors and fidelity errors compared with uniform binning. We believe fewer bins help capture overall correlations by introducing less noise, which benefits deep learning tasks. However, query tasks and fidelity are more sensitive to exact values, and more bins lead to higher accuracy. Two exceptions are Private-GSD and GEM, where PrivTree outperforms uniform binning. We hypothesize that this is because PrivTree reduces the dimensionality of variables, thereby improving the convergence. These results do not completely align with observations in Tao et al.'s work [60]. We would propose three possible reasons. Firstly, Higgs-small dataset is more value-rich than the datasets used in their work, which can lead to high sensitivity to binning methods. Secondly, the evaluation metrics are different, while ours focus more on measurements on higher-dimensional marginals (e.g., 3-way marginals). This can influence the results. Finally, the investigated algorithms vary in the two works, which may lead to different comparison conclusions. Another unusual finding is that under PrivTree binning, AIM shows a significantly worse runtime. We believe this is because smaller domain sizes will cause AIM to allocate budget to more marginals. This may lead to large cliques, slowing down the execution speed of PGM.

We can conclude that different discretization methods have their own advantages and weaknesses. It is important to choose an appropriate method for different algorithms. For example, for those methods based on generative networks, PrivTree could be a potentially better preprocessing method because it can significantly decrease the dimension of models by reducing domain complexity. For PGM and GUM, uniform binning could be a better choice due to their stronger ability to fit marginals with large domains.

Category Merging Ablation. Finally, we briefly discuss the necessity of introducing a fixed merging threshold. We compare the maximum domain size of preprocessed categorical attributes under different fixed merging thresholds in Figure 7. We can conclude that domain size cannot be reduced efficiently under large ε if we do not apply a fixed merging threshold (0.0%). Furthermore, we may over-merge categories if this threshold is large (e.g., 1%), which may cause potential synthesis errors.

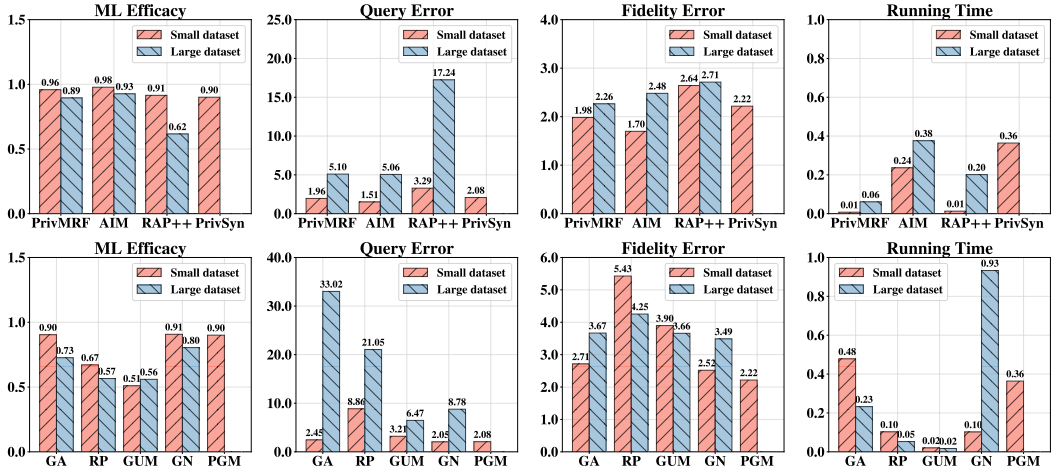


Fig. 8. Average scaled evaluations for different marginal selection modules (first row) and synthesis modules (second row) on different datasets. We use PGM as the common data synthesizer and PrivSyn as the common marginal selector. The running time is scaled by min-max linear normalization, and others are scaled using ground truth values provided in Table 4. The small datasets include ACSincome and ACSemploy, whereas the large datasets consist of Bank, Higgs-small, and Loan.

9.3 Module Comparison

In this subsection, we decompose our experiments into two parts: one focusing on different feature selection algorithms and the other on different synthesis modules. In other words, we fix either the selection or the synthesis approach and then evaluate how different algorithms perform under that fixed condition. Since some features are not compatible with other synthesis methods (e.g., random Fourier features), we only focus on marginal-based methods, namely PrivSyn, RAP++, Private-GSD, AIM, PrivMRF, and GEM.

To make clearer comparisons, we divide the datasets into two groups. The first group, ACSincome and ACSemploy, is relatively low-dimensional with moderately sized attributes' domains. We call them "small datasets" for description simplicity. The second group, Bank, Higgs-small, and Loan, is higher-dimensional or has larger attribute domains, which are called "large datasets". To make a better comparison, we scale each metric and display the average performance on these two groups of datasets under different ε.

Comparison of Feature Selection Modules. Since some algorithms use the same or similar marginal selection methods, or have not specified them, we consider four marginal selection methods included in previous work. They are: PrivSyn selection, PrivMRF selection, RAP++ selection, and AIM selection. The synthesis processes for these four selection methods are all set to PGM. Figure 8 show the average scaled performances.

Among the selection methods, PrivSyn and RAP++ show weaker fitting utility, consistent with our analysis. PrivSyn fails to use intermediate information during selection, reducing its ability to capture necessary data features and leading to excessive selections. This also overwhelms PGM on larger datasets due to high memory demands. Similarly, RAP++ employs the Gumbel mechanism to select multiple marginals per iteration, without fully accounting for intermediate results. It also decreases the number of synthesis rounds required, thereby accelerating the algorithm's execution. Moreover, its selection criterion focuses solely on marginal query error, ignoring noise error, which is unbalanced and potentially influences the algorithm's performance.

The utility performances of AIM and PrivMRF, are very similar since both perform a single marginal refinement after updating the intermediate information. A notable observation is that PrivMRF has a shorter running time than AIM. We attribute this to the well-designed initial marginal set in PrivMRF, which decreases the need for further refinement and speeds up the total algorithm.

Comparison of Synthesis Modules. We compare the synthesis algorithms of all six algorithms: GUM, PGM, relaxed projection (RP), genetic algorithm (GA), and generative network (GN). PGM is employed by both AIM and PrivMRF. To avoid multiple fitting steps from adaptive selection and control the running time, we use the PrivSyn selection algorithm as the common marginal selector. The results are shown in Figure 8. GUM and PGM are conducted on CPU, and the remaining works are on GPU.

Among these methods, PGM and the generative network achieve the best overall accuracy and stability in fitting features on small datasets. PGM benefits from its expressive structure, which, as indicated by Equation (2), infers marginals without refitting existing ones, thus minimizing compounding errors. However, the PGM's fitting process is quite slow, due to the densely selected marginals by PrivSyn, causing it to fail to deal with large datasets. Indeed, PGM is most efficient when coupled with a marginal selection procedure that is designed to make the resulting graphical model "tree-like" [7, 39, 40]. The generative network's strong representational capacity also supports accurate feature fitting on small datasets. However, when it turns to larger and more complex datasets, its superiority in efficiency and utility will be diminished due to the greater difficulty of training high-dimensional models.

In contrast, GUM, the genetic algorithm, and relaxed projection demonstrate weaker fitting capabilities. GUM refines marginals individually, overlooking global correlations. However, it shows a superiority in running time, aligning with our analysis. The genetic algorithm's reliance on randomness makes it unsuitable for complex, high-dimensional datasets with large attribute domains under time constraints, leading to poor performance on large datasets. Relaxed projection suffers similarly. In high-dimensional spaces with extensive attributes' domains, the encoded data dimension becomes excessively large, complicating optimization and making it harder to converge to the optimal point, and finally resulting in poor performance.

10 Conclusion and Discussion

Data synthesis under differential privacy remains a critical and active area of research. Despite a growing number of methods being proposed, the field still lacks a fair and comprehensive comparative analysis. In this paper, we conduct analysis, comparison, and evaluation of several methods on a unified framework. The primary findings of our study are as follows:

- *A significant trade-off exists between utility and efficiency, resulting in no single algorithm dominating the others.* deep learning-based methods, though highly efficient, generally exhibit inferior performance compared to traditional statistical methods. Nonetheless, under current implementations, some statistical methods, such as AIM and PrivMRF, are often time-consuming, which is a trade-off for superior utility.
- *Data preprocessing strategy is important but algorithm-dependent.* Data preprocessing is crucial in reducing algorithm complexity without introducing too much error. The choice of data preprocessing methods should align with the working principles of the algorithm. Appropriate preprocessing techniques can enhance both the efficiency and effectiveness of the algorithms, whereas unsuitable methods may hinder their performance.
- *Improvement on data synthesis module remains a promising direction.* We find that some feature selection methods outperform others, but existing synthesis modules exhibit significant drawbacks in different ways. For example, even though PGM demonstrates outstanding utility, it cannot

Table 9. Estimated cost (US dollar) of running different algorithms.

Method	PrivSyn	PrivMRF	AIM	RAP++	Private-GSD	GEM	DP-MERF	TabDDPM
Total Cost	$ 0.2	$ 1.1	$ 5.7	$ 68.1	$ 17.0	$ 0.7	$ 0.1	$ 2.1

work on densely selected marginals (due to high memory requests and long execution time), while generative networks face significant challenges when dealing with high-dimensional data. There remains substantial room for improvement in developing algorithms that can efficiently, reliably, and accurately generate data.

Our work not only sheds light on the strengths and limitations of current methods but also identifies several promising directions for future research. These include tailoring preprocessing strategies to algorithmic needs, developing more effective feature selection techniques, and improving the utility of deep learning models or the efficiency of some statistical methods by more efficient implementation or GPU acceleration. In addition to these research opportunities, there are still certain other aspects, though not investigated in this work, worth exploring. For example, beyond the time efficiency perspective, dollar cost is a more realistic factor in algorithm selection. Based on our experimental setups and hardware, we roughly set the cost at $0.8/hr for GPU usage, and $0.2/hr for CPU usage (the actual prices may vary due to many factors such as dataset complexity, which can affect our selection of hardware, and price fluctuations of cloud service [4, 50, 51, 54], etc.). We summarize the total price consumption of all algorithms during the overall comparison in Table 9. We can observe that those methods with lower time efficiency tend to incur higher costs.

Finally, we discuss the scalability of our framework. It offers a broader perspective for analyzing existing synthesis algorithms, but we also need to recognize its limitations. For example, we regard gradients as features to explain those methods based on DP-SGD. But the gradient feature itself is not interpretable enough. Besides, some “model-level DP" methods may fall outside our three-step workflow. For instance, PATE-GAN [28] uses multiple teacher discriminators to achieve DP and trains the student discriminator on them to further guide the data generator. In this case, our framework may lack corresponding explanatory power and require further extension to explain these methods.

Acknowledgments

We thank anonymous reviewers for their valuable comments. This paper was partially supported by NSF CNS-2220433, NSF OAC-2319988, and a CCI award.

References

[1] 2016. Higgs particle dataset. https://www.openml.org/search?type=data&status=active&id=4532
[2] 2020. Loan dataset. https://www.kaggle.com/datasets/devanshi23/loan-data-2007-2014
[3] Sergul Aydore, William Brown, Michael Kearns, Krishnaram Kenthapadi, Luca Melis, Aaron Roth, and Ankit A Siva. 2021. Differentially private query release through adaptive projection. In *International Conference on Machine Learning*. PMLR, 457–467.
[4] Microsoft Azure. 2025. Virtual Machines Pricing. https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/.
[5] Andrés F. Barrientos, Alexander Bolton, Tom Balmat, Jerome P. Reiter, John M. de Figueiredo, Ashwin Machanavajjhala, Yan Chen, Charley Kneifel, and Mark DeLong. 2018. Providing Access to Confidential Research Data Through Synthesis and Verification: An Application to Data on Employees of the U.S. Federal Government. arXiv:1705.07872 [stat.AP] https://arxiv.org/abs/1705.07872
[6] Vincent Bindschaedler, Reza Shokri, and Carl A. Gunter. 2017. Plausible Deniability for Privacy-Preserving Data Synthesis. arXiv:1708.07975 [cs.CR] https://arxiv.org/abs/1708.07975
[7] Kuntai Cai, Xiaoyu Lei, Jianxin Wei, and Xiaokui Xiao. 2021. Data synthesis via differentially private markov random fields. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2190–2202.

[8] Mark Cesar and Ryan Rogers. 2021. Bounding, concentrating, and truncating: Unifying privacy loss composition for data analytics. In *Algorithmic Learning Theory*. PMLR, 421–457.
[9] Chris Chatfield. 2018. *Introduction to multivariate analysis*. Routledge.
[10] Kai Chen, Xiaochen Li, Chen Gong, Ryan McKenna, and Tianhao Wang. 2025. Benchmarking differentially private tabular data synthesis. *arXiv preprint arXiv:2504.14061* (2025).
[11] Harald Cramér. 1999. *Mathematical methods of statistics*. Vol. 9. Princeton university press.
[12] Teddy Cunningham, Graham Cormode, and Hakan Ferhatosmanoglu. 2021. Privacy-Preserving Synthetic Location Data in the Real World. In *17th International Symposium on Spatial and Temporal Databases (SSTD '21)*. ACM, 23–33. doi:10.1145/3469830.3470893
[13] Travis Dick, Cynthia Dwork, Michael Kearns, Terrance Liu, Aaron Roth, Giuseppe Vietri, and Zhiwei Steven Wu. 2023. Confidence-ranked reconstruction of census microdata from published statistics. *Proceedings of the National Academy of Sciences* 120, 8 (Feb. 2023). doi:10.1073/pnas.2218605120
[14] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2021. Retiring Adult: New Datasets for Fair Machine Learning. *Advances in Neural Information Processing Systems* 34 (2021).
[15] Tim Dockhorn, Tianshi Cao, Arash Vahdat, and Karsten Kreis. 2023. Differentially Private Diffusion Models. arXiv:2210.09929 [stat.ML] https://arxiv.org/abs/2210.09929
[16] Yuntao Du and Ninghui Li. 2024. Towards principled assessment of tabular data synthesis algorithms. *arXiv preprint arXiv:2402.06806* (2024).
[17] Liyue Fan. 2020. A survey of differentially private generative adversarial networks. In *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*, Vol. 8.
[18] Mei Ling Fang, Devendra Singh Dhami, and Kristian Kersting. 2022. Dp-ctgan: Differentially private medical data generation using ctgans. In *International conference on artificial intelligence in medicine*. Springer, 178–188.
[19] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. 2015. Dual Query: Practical Private Query Release for High Dimensional Data. arXiv:1402.1526 [cs.DS] https://arxiv.org/abs/1402.1526
[20] Georgi Ganev, Meenatchi Sundaram Muthu Selva Annamalai, Sofiane Mahiou, and Emiliano De Cristofaro. 2025. The Importance of Being Discrete: Measuring the Impact of Discretization in End-to-End Differentially Private Synthetic Data. arXiv:2504.06923 [cs.CR] https://arxiv.org/abs/2504.06923
[21] Chen Gong, Kecen Li, Zinan Lin, and Tianhao Wang. 2025. DPImageBench: A Unified Benchmark for Differentially Private Image Synthesis. *arXiv preprint arXiv:2503.14681* (2025).
[22] Chen Gong, Kecen Li, Zinan Lin, and Tianhao Wang. 2025. DPImageBench: A Unified Benchmark for Differentially Private Image Synthesis. arXiv:2503.14681 [cs.CR] https://arxiv.org/abs/2503.14681
[23] Frederik Harder, Kamil Adamczewski, and Mijung Park. 2021. Dp-merf: Differentially private mean embeddings with randomfeatures for practical privacy-preserving data generation. In *International conference on artificial intelligence and statistics*. PMLR, 1819–1827.
[24] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. *Advances in neural information processing systems* 25 (2012).
[25] Moritz Hardt and Guy N Rothblum. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st annual symposium on foundations of computer science*. IEEE, 61–70.
[26] Yuzheng Hu, Fan Wu, Qinbin Li, Yunhui Long, Gonzalo Munilla Garrido, Chang Ge, Bolin Ding, David Forsyth, Bo Li, and Dawn Song. 2023. SoK: Privacy-Preserving Data Synthesis. arXiv:2307.02106 [cs.CR] https://arxiv.org/abs/2307.02106
[27] IPUMS. 2025. IPUMS: Integrated Public Use Microdata Series. https://www.ipums.org/. https://www.ipums.org/
[28] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. 2018. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*.
[29] Adam Kalai and Santosh Vempala. 2005. Efficient algorithms for online decision problems. *J. Comput. System Sci.* 71, 3 (2005), 291–307.
[30] Rashid Hussain Khokhar, Benjamin CM Fung, Farkhund Iqbal, Khalil Al-Hussaeni, and Mohammed Hussain. 2023. Differentially private release of heterogeneous network for managing healthcare data. *ACM Transactions on Knowledge Discovery from Data* 17, 6 (2023), 1–30.
[31] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2023. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*. PMLR, 17564–17579.
[32] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.
[33] Haoran Li, Li Xiong, and Xiaoqian Jiang. 2014. Differentially private synthesization of multi-dimensional data using copula functions. In *Advances in database technology: proceedings. International conference on extending database technology*, Vol. 2014. NIH Public Access, 475.
[34] Kecen Li, Chen Gong, Zhixiang Li, Yuzhong Zhao, Xinwen Hou, and Tianhao Wang. 2024. {PrivImage}: Differentially Private Synthetic Image Generation using Diffusion Models with {Semantic-Aware} Pretraining. In *33rd USENIX*

Security Symposium (USENIX Security 24). 4837–4854.
[35] Ninghui Li, Zhikun Zhang, and Tianhao Wang. 2021. DPSyn: Experiences in the NIST Differential Privacy Data Synthesis Challenges. arXiv:2106.12949 [cs.CR] https://arxiv.org/abs/2106.12949
[36] Terrance Liu, Jingwu Tang, Giuseppe Vietri, and Steven Wu. 2023. Generating private synthetic data with genetic algorithms. In *International Conference on Machine Learning*. PMLR, 22009–22027.
[37] Terrance Liu, Giuseppe Vietri, and Steven Z Wu. 2021. Iterative methods for private synthetic data: Unifying framework and new methods. *Advances in Neural Information Processing Systems* 34 (2021), 690–702.
[38] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing error of high-dimensional statistical queries under differential privacy. *Proceedings of the VLDB Endowment* 11, 10 (June 2018), 1206–1219. doi:10.14778/3231751.3231769
[39] Ryan McKenna, Gerome Miklau, and Daniel Sheldon. 2021. Winning the NIST Contest: A scalable and general approach to differentially private synthetic data. *arXiv preprint arXiv:2108.04978* (2021).
[40] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. 2022. Aim: An adaptive and iterative mechanism for differentially private synthetic data. *arXiv preprint arXiv:2201.12677* (2022).
[41] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*. PMLR, 4435–4444.
[42] Ilya Mironov. 2017. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 263–275.
[43] Ilya Mironov, Kunal Talwar, and Li Zhang. 2019. R\'enyi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530* (2019).
[44] S. Moro, P. Rita, and P. Cortez. 2014. Bank Marketing. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5K306.
[45] Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2018. How to Use Heuristics for Differential Privacy. arXiv:1811.07765 [cs.LG] https://arxiv.org/abs/1811.07765
[46] NIST. 2018. 2018 Differential Privacy Synthetic Data Challenge. Available at https://www.nist.gov/ctl/pscr/open-innovation-prize-challenges/past-prize-challenges/2018-differential-privacy-synthetic.
[47] NIST. 2020. 2020 Differential Privacy Temporal Map Challenge. Available at https://www.nist.gov/ctl/pscr/open-innovation-prize-challenges/past-prize-challenges/2020-differential-privacy-temporal.
[48] Wei Pang, Masoumeh Shafieinejad, Lucy Liu, and Xi He. 2024. ClavaDDPM: Multi-relational Data Synthesis with Cluster-guided Diffusion Models. *arXiv preprint arXiv:2405.17724* (2024).
[49] Gabriel Peyré, Marco Cuturi, et al. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning* 11, 5-6 (2019), 355–607.
[50] Google Cloud Platform. 2025. Google Cloud Compute Engine Pricing. https://cloud.google.com/compute/all-pricing.
[51] Runpod. 2025. RTX 6000 Ada Generation Pricing. https://www.runpod.io/gpu-models/rtx-6000-ada.
[52] Alexandre Sablayrolles, Yue Wang, and Brian Karrer. 2023. Privately generating tabular data using language models. arXiv:2306.04803 [cs.LG] https://arxiv.org/abs/2306.04803
[53] S Sangeetha, G Sudha Sadasivam, and Ayush Srikanth. 2022. Differentially private model release for healthcare applications. *International Journal of Computers and Applications* 44, 10 (2022), 953–958.
[54] Amazon Web Services. 2025. EC2 On-Demand Pricing. https://aws.amazon.com/cn/ec2/pricing/on-demand/.
[55] Mani Srivastava and Moustafa Alzantot. 2019. Differentially Private Dataset Release using Wasserstein GANs. https://github.com/nesl/nist_differential_privacy_synthetic_data_challenge. Accessed 2023-xx-xx.
[56] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. 2022. Synthetic Data – Anonymisation Groundhog Day. arXiv:2011.07018 [cs.LG] https://arxiv.org/abs/2011.07018
[57] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2018. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. arXiv:1712.06567 [cs.NE] https://arxiv.org/abs/1712.06567
[58] Arun Sai Suggala and Praneeth Netrapalli. 2020. Online non-convex learning: Following the perturbed leader is optimal. In *Algorithmic Learning Theory*. PMLR, 845–861.
[59] Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert Schapire. 2016. Efficient algorithms for adversarial contextual learning. In *International Conference on Machine Learning*. PMLR, 2159–2168.
[60] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. 2022. Benchmarking Differentially Private Synthetic Data Generation Algorithms. arXiv:2112.09238 [cs.CR] https://arxiv.org/abs/2112.09238
[61] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. 2019. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 0–0.
[62] Toan V Tran and Li Xiong. 2024. Differentially Private Tabular Data Synthesis using Large Language Models. *arXiv preprint arXiv:2406.01457* (2024).

[63] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[64] Giuseppe Vietri, Cedric Archambeau, Sergul Aydore, William Brown, Michael Kearns, Aaron Roth, Ankit Siva, Shuai Tang, and Steven Z Wu. 2022. Private synthetic data for multitask learning and marginal queries. *Advances in Neural Information Processing Systems* 35 (2022), 18282–18295.

[65] Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Zhiwei Steven Wu. 2020. New Oracle-Efficient Algorithms for Private Synthetic Data Release. arXiv:2007.05453 [cs.LG] https://arxiv.org/abs/2007.05453

[66] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739* (2018).

[67] Mengmeng Yang, Chi-Hung Chi, Kwok-Yan Lam, Jie Feng, Taolin Guo, and Wei Ni. 2024. Tabular Data Synthesis with Differential Privacy: A Survey. *arXiv preprint arXiv:2411.03351* (2024).

[68] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. 2021. Opacus: User-Friendly Differential Privacy Library in PyTorch. *arXiv preprint arXiv:2109.12298* (2021).

[69] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 1–41.

[70] Jun Zhang, Xiaokui Xiao, and Xing Xie. 2016. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 international conference on management of data*. 155–170.

[71] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. 2021. {PrivSyn}: Differentially private data synthesis. In *30th USENIX Security Symposium (USENIX Security 21)*. 929–946.

Received April 2025; revised July 2025; accepted August 2025