Why LLMs Are Bad at Synthetic Table Generation (and what to do about it)

Shengzhe Xu
Computer Science
Virginia Tech
Alexandria, VA, USA
shengzx@vt.edu

Cho-Ting Lee
Computer Science
Virginia Tech
Alexandria, VA, USA
choting@vt.edu

Mandar Sharma
Computer Science
Virginia Tech
Alexandria, VA, USA
mandarsharma@vt.edu

Raquib Bin Yousuf
Computer Science
Virginia Tech
Alexandria, VA, USA
raquib@vt.edu

Nikhil Muralidhar
Computer Science
Stevens Institute of Technology
Hoboken, NJ, USA
nmurali1@stevens.edu

Naren Ramakrishnan
Computer Science
Virginia Tech
Alexandria, VA, USA
naren@cs.vt.edu

Abstract

Synthetic data generation is integral to ML pipelines, e.g., to augment training data, replace sensitive information, and even to power advanced platforms like DeepSeek. While LLMs fine-tuned for synthetic data generation are gaining traction, synthetic table generation—a critical data type in business and science—remains underexplored compared to text and image synthesis. This paper shows that LLMs, whether used as-is or after traditional fine-tuning, are inadequate for generating synthetic tables. Their autoregressive nature, combined with random order permutation during fine-tuning, hampers the modeling of functional dependencies and prevents capturing conditional mixtures of distributions essential for real-world constraints. We demonstrate that making LLMs permutation-aware can mitigate these issues. Our code and data are publicly hosted [1].

1 Introduction

Large language models (LLMs) have found applicability in a rich diversity of domains, well beyond their original roots [1, 4, 29, 32, 40, 42, 48]. As so-called foundation models [22] they have been shown to be re-targetable to a variety of downstream tasks. Our focus here is to view LLMs as raw synthetic tabular data generators rather than as supporting an analysis or discovery task. Arguably, LLMs are adept at synthetic generation of text, images, videos, code, documentation, and many other modalities. The use of LLMs to generate tabular data is quite understudied.

Such synthetic tabular data generation is integral to ML pipelines, e.g., too augment training data, replace sensitive information, and even to power advanced platforms like DeepSeek [17, 24]. However, the unique characteristics of tabular data manifest as challenges in an LLM-based generative context. The most popular incarnations of language models are auto-regressive models, e.g., LLama [48], GPT-x [32], DeepSeek [17, 24], wherein each word or token is generated conditional on past tokens in a sequential manner using attention models. In a synthetic data context, each 'sentence' typically represents a row of tabular data, and each 'word' corresponds to an attribute in that row. The previous state-of-the-art models (GReaT [3]), has advocated the use of random feature orders but as we show in Table 1, when fine-tuning is done with random feature orders, key relationships are often not captured or, worse, violated. In particular, with tabular data, there are numerous functional dependencies at play and as a result, generating tokens in random orders is bound to cause violations. There is thus an **'impedance mismatch' between autogressive LLMs and synthetic data generation.**

The main contributions of this paper are:

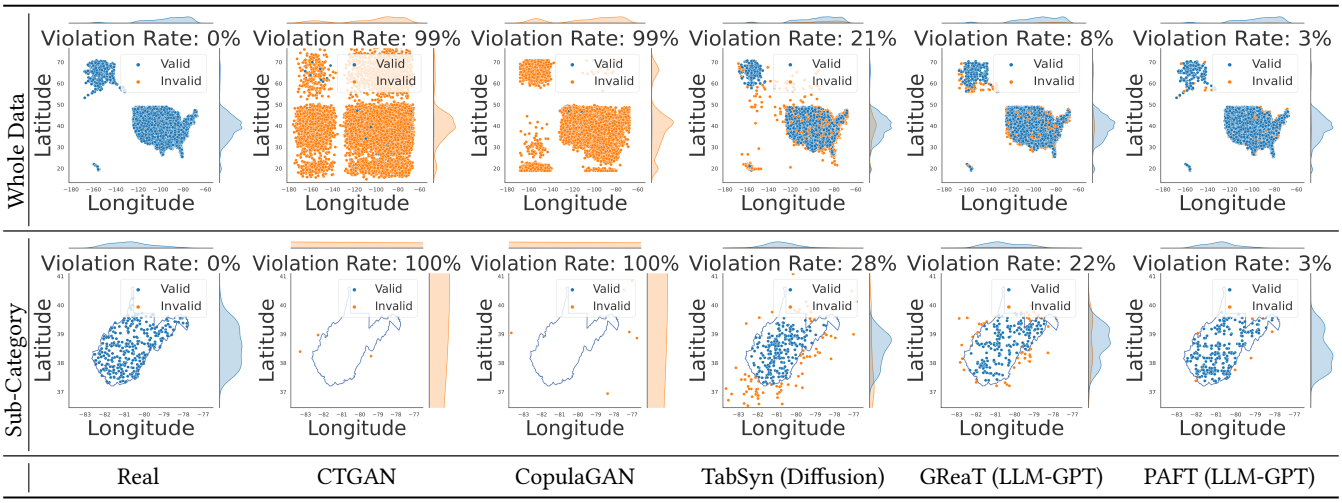
- We highlight an important deficiency with using LLMs for synthetic tabular data generation and explore the performance of many state-of-the-art generation models in the context of composite and multi-category tabular schema.
- We inject knowledge of pre-existing functional relationships among columns into the autoregressive generation process, so that the generated synthetic data respects more real constraints. In particular, we present a taxonomy of functional dependencies (FDs) whose discovery and organization into a column dependence graph supports their incorporation into the LLM fine-tuning process via a permutation function, leading to our approach dubbed Permutation-aided Fine-tuning.(PAFT).
- We evaluate the performance of PAFT on a range of datasets featuring a diverse mix of attribute types, functional dependencies, and complex relationships. Our results demonstrate that PAFT is the state-of-the-art in reproducing underlying relationships in generated synthetic data.
- Finally, we demonstrate through rigorous experiments that relying just on standard univariate distribution, bivariate correlation, and even the evaluation of downstream machine learning models (which primarily focuses on predicting a single column in a dataset) is grossly insufficient for assessing the quality of synthetic data and propose systematic remedies like measuring violation rates of known domain rules.

2 Related Work

Tabular data synthesis and representation learning for tables have been extensively studied [12, 13, 15, 27, 28, 35, 36, 50, 51, 55]. One key theme has been the discovery of functional dependencies (FDs) which has been studied by the data mining community from both

[1] https://github.com/ShengzheXu/Permutation-aided-Fine-tuning

Table 1: Comparison of multiple synthetic data generation approaches. The second row showcases the state of West Virginia (WV), which is a subset of the whole data. In the figures, the solid line represents the official border of West Virginia and the Blue and Orange colors indicate the legal and illegal samples in the synthetic data respectively.



theoretical and application points of view [5, 26, 30, 37, 57, 60]. We carve out related work into two sections, for our purposes: pre-LLM (or non-LLM) approaches for synthetic table generation which continue to hold the mainstay, and LLM approaches.

Lei et al. [52] proposed CTGAN where rows are independent of each other; a conditional GAN architecture ensures that the dependency between columns is learned. Tabsyn [55] showcased remarkable advancements in joint-distribution learning via a VAE plus diffusion approach, surpassing previous models of similar lineage, in terms of distributional correlation measures and machine learning efficiency. DoppelGanger [23] uses a combination of an RNN and a GAN to incorporate temporal dependencies across rows but this method has been tested in traditional, low-volume settings such as Wikipedia daily visit counts. For high-volume applications, STAN [54] utilizes a combination of a CNN and Gaussian mixture neural networks to generate synthetic network traffic data. GraphDF [6] conducts multi-dimensional time series forecasting. GOGGLE [25] employs a generative modeling method for tabular data by learning relational structures.

The use of language models (LLMs) for tabular data generation is still underexplored. Most modern LLMs are based on the transformer architecture [49] with parameters ranging from few millions to billions [18], and researchers have developed creative ways to harness LLMs in traditional machine learning and data contexts. LIFT [10] initially transforms a table row into a sentence, such as 'An Iris plant with sepal length 5.1cm, sepal width 3.5cm', and employs an LLM as a learning model for table classification, regression, and generation tasks. GReaT [3] utilizes a GPT-2 model that has been fine-tuned using a specific corpus for synthetic data generation. They also show that even small-scale models such as Distill-GPT [39] have the potential for synthetic data generation [3]. These models are specially viable for tabular generation given the lower compute costs of aligning smaller models to large and varied tabular datasets. A general benefit of utilizing LLMs is the promise of eliminating customized preprocessing pipelines.

A theme that will be pertinent to our work is the idea of feature ordering (FO) for tabular data generation which has been investigated from multiple angles [11, 41, 45, 61]. There are also several approaches (e.g., [9, 20]) that synthesize, discover, or aggregate features from relational databases, leveraging order information when possible, for use in machine learning pipelines. It worth noting that even in the LLM community, the task of context sorting for LLM prompting is not trivial and has gained significant attention lately [8].

3 Challenges to Synthetic Table Generation in the Current LLM Paradigm

Admittedly, the sure-fire way to check if a dataset has been faithfully modeled is to see if the joint distribution of its features is captured accurately. Most current tests of synthetic data generation quality focus on fidelity to single-column distributions, or to multi-column distributions. For multiple variables, measures such as machine learning efficiency (MLE) [3, 10, 52, 55] are frequently used to construct classifiers or regressors.

A key lesson from probabilistic graphical model research [21] is that factorizing joint distributions into products of conditionals (e.g., Bayesian networks) dramatically helps reduce the number of parameters necessary to capture the underlying data characteristics. A similar lesson from database research [16] is that modeling functional dependencies (FDs) in data helps reduce redundancy in modeling and storage. These lessons, i.e., that **order matters**, continue to apply in the LLM era but are not internalized in our prompt ordering, fine-tuning, or evaluation methodologies. Other researchers have noted the importance of ordering in LLMs [8, 38] but this lesson has not been leveraged to improve synthetic table generation by LLMs. In the absence of leveraging good feature orders, existing approaches either focus on 'one order', 'no order',

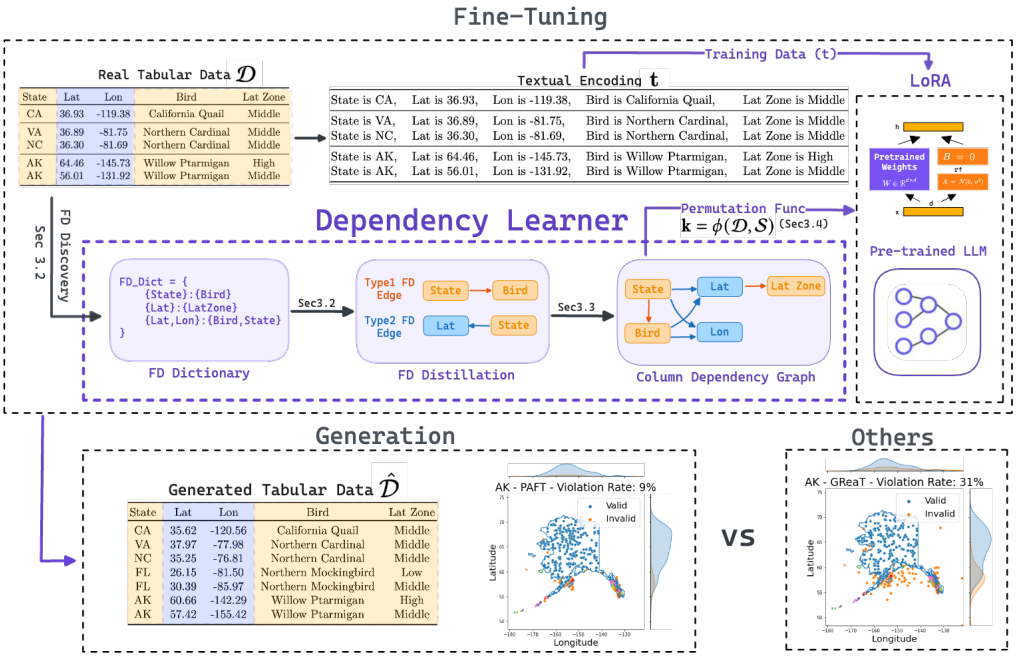


Figure 1: Overview of the proposed Permutation-Aided Fine-tuning (PAFT) approach.

or 'all orders'. All of these approaches severely limit the quality of generated synthetic data.

As an example, Table 1 presents a case study on using models to generate synthetic data of locations in various states of the USA. The data is of the form (state, latitude, longitude) where the attributes adhere to the FD: {latitude, longitude} $\longrightarrow$ state. Existing methods can generate satisfactory univariate distributions (as shown in the border of the top row plots) but fail to capture the joint distribution (center of the top row plots) and conditional distributions across subcategories (bottom row plots).

In summary, feature ordering can be both a nuisance and a gift. It is a nuisance because it demands additional constraints to be modeled. It can be a gift because it suggests ways to sequentially generate data even by autoregressive LLMs. Our proposed approach (PAFT) aims to achieve an optimal permutation order for fine-tuning LLMs.

Figure 1 shows an overview of the proposed permutation aided fine-tuning approach (PAFT). A typical workflow is 1) Textual Encoding (Section 4.1) 2) Functional Dependency (FD) discovery (Section 4.2) 3) FD Distillation (Section 4.2) and 4) Feature Order Permutation Optimization (Section 4.3). Our fine-tuning and sampling strategy is explained in Section 4.4.

4 PAFT: Permutation-Aided Fine-Tuning

Problem Setup. Let $\mathcal{D}$ represent a table with n rows (i.e., records) and m columns (i.e., attributes a.k.a schema). Let each record be represented by vector $\mathbf{x}_i$ and further let x_{ij} represent the element value of the jth attribute of record $\mathbf{x}_i$. Hence each row $\mathbf{x}_i \in \mathcal{D}$ represents an individual record and each column $\mathbf{x}_{(:,j)} \sim \mathcal{X}_j$ can be considered sampled from a random variable $\mathcal{X}_j$ that governs the distribution of attribute j. Finally, let $i \in [1..n]$ and $j \in [1..m]$. Realistically, tabular data $\mathcal{D}$ is frequently a mixture of categorical and continuous attributes, hence each $\mathcal{X}_j$ can be a categorical or continuous random variable. If $\mathcal{A} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m\}$ represents the collection of random variables, then the table generation process aims to sample from a joint distribution $\mathbb{P}(\mathcal{A}) = \mathbb{P}(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_m)$. This joint distribution is usually a complex, high-dimensional distribution and, most importantly, unknown. The goal of learning an effective tabular data generator $p_\theta(\cdot)$ is to enable $p_\theta(\cdot)$ to learn a *faithful* approximation $\mathbb{P}(\mathcal{A}|\mathcal{D})$ of the data generation process distribution $\mathbb{P}(\mathcal{A})$ using the data sample $\mathcal{D}$ such that $\mathbb{P}(\mathcal{A}|\mathcal{D}) \approx \mathbb{P}(\mathcal{A})$. Once such an effective model $p_\theta(\mathcal{D})$ is trained, it can be employed to generate large volumes of seemingly *realistic* synthetic data $\hat{\mathcal{D}} \sim \mathbb{P}(\mathcal{A}|\mathcal{D})$.

4.1 Tabular Data Generation with LLMs

While training $p_\theta(\cdot)$, it is usually assumed that all records $\mathbf{x}_i \in \mathcal{D}$ are independent. Generating new data samples $\hat{\mathbf{x}}_i \in \hat{\mathcal{D}}$ can be done in various ways (e.g., see [52, 53, 59]) which aim to directly estimate the joint distribution $\mathbb{P}(\mathcal{A})$ or, as is done here in PAFT, where $\mathbb{P}(\mathcal{A})$ is estimated by an autoregressive LLM based generative process, as a product of multiple conditional densities governed by the input ordering.

Autoregressive LLM models are pre-trained to maximize the likelihood of *target* token $x_{ij} \in \mathcal{D}$, conditioned upon the autoregressive context $\mathbf{x}_{(i,1:j-1)} \in \mathcal{D}$ where $\mathcal{D}$ is the training corpus comprising a large amount of textual data (in the pre-training context). Eq. 1 defines the general training criterion of LLM training using the self-supervised next-token prediction task with 'w' denoting the context length.

$$\mathcal{L}(\theta; \mathcal{D}) = -\sum_{\mathbf{x}_i \in \mathcal{D}} \sum_{j=1}^{w} \log \mathbb{P}(x_i | \mathbf{x}_{(i,1:j-1)}). \tag{1}$$

The generation of a single instance (i.e., database record) $\mathbf{x}_i \in \mathcal{D}$ is given by Eq. 2:

$$\mathbb{P}(\mathbf{x}_i) = \mathbb{P}(x_{i,1}, ..., x_{i,m}) \simeq \prod_{j=1}^{m} \mathbb{P}(x_{i,j} | x_{i,1}, .., x_{i,j-1}). \tag{2}$$

Specifically, each database record is generated as a product of conditional distributions.

Input Encoding. To support the processing of our records $\mathbf{x}_i \in \mathcal{D}$ by a pre-trained LLM, we adopt the following encoding:

$$\begin{aligned} t_{i,j} &= [c_j, \text{`is'}, x_{i,j}, \text{`,'}], i \in \{1, .., n\}, j \in \{1, .., m\}, \\ \mathbf{t}_i &= [t_{i,1}, t_{i,2}, .., t_{i,m}], i \in \{1, .., n\}. \end{aligned} \tag{3}$$

In Eq. 3, c_j represents the attribute name of the jth database column while $x_{i,j}$ represents the actual value of the jth column for the ith record. Further, we can assume we have a mechanism to obtain a *feature order permutation* $\mathbf{k}$ to govern the order of the attributes in $\mathbf{t}_i$, such that $\mathbf{t}_i(\mathbf{k}) = [t_{i,k_1}, t_{i,k_2}, .., t_{i,k_m}]$ (where $i \in \{1, .., n\}, k_j \in \{1, .., m\}$.), represents the same record but with the attribute order governed by the permutation $\mathbf{k}$, This definition admits the random feature order as a special case in which $\mathbf{k}$ is a random permutation.

Since we consider autoregressive LLM-based generative models, employing the chain rule to sequentially produce each column of a table record $\mathbf{t}_i$, we can view each generation step as *approximating* the joint distribution of the table columns as a product of conditional distributions (i.e., $\mathbb{P}(t_{i,1}, ..., t_{i,m}) \simeq \prod_{j=1}^{m} \mathbb{P}(t_{i,j} | t_{i,1}, .., t_{i,j-1})$). However, as the number of columns increases and the relationships between columns get more conditional, the likelihood of encountering training and sampling bias due to class imbalance also rises [52]. To minimize such adverse effects, we can consider injecting knowledge of the pre-existing functional relationships among columns, to govern the autoregressive generation process. To infer such functional relationships, we leverage a learned dependency graph derived from functional dependency (FD) relations which enables us to effectively determine the appropriate training and sampling sequence. This, in turn, allows us to alleviate potential biases during training by establishing a *generation curriculum* leading to improved estimation accuracy of the joint distribution $\mathbb{P}(\mathbf{t}_i)$ in auto-regressive prediction $\mathbb{P}(t_{i,k_1}, ..., t_{i,k_m}) \simeq \prod_{j=1}^{m} \mathbb{P}(t_{i,k_j} | t_{i,k_1}, .., t_{i,k_{j-1}})$, where the ordering $t_{i,k_1} \ldots, t_{i,k_m}$ is obtained by a feature order permutation function $\mathbf{k} = \phi(\mathcal{D}, \mathcal{S})$. We detail the requisite background and design of $\phi(\mathcal{D}, \mathcal{S})$ in sections 4.2 and 4.3.

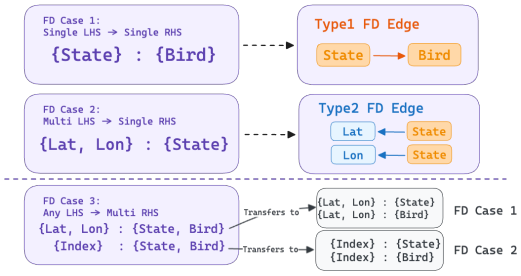
4.2 Discovery and Distillation of Functional Dependencies (FD)

A functional dependency (FD) is a relationship R in schema S that exists when a subset of attributes $A \subset S$ uniquely determines another subset $B \subset S$ of attributes. We succinctly represent an FD as $R : A \rightarrow B$ which specifies that B is functionally dependent on A.

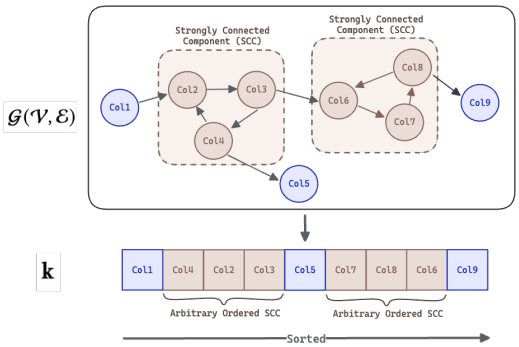
Definition 1 (Schema-Level FD). *With A, B being two disjoint subsets of the schema (columns) of table $\mathcal{D}$, a schema-level FD F associated with $\mathcal{D}$ has the form: $F : A \rightarrow B$.*

We leverage FD discovery techniques to govern the order of the autoregressive data generation process in PAFT . A large body of research from the database literature on FD discovery [33, 34, 57] can be leveraged in PAFT , including methods that account for noisy FDs [57]. In this work, we focus on leveraging schema-level FDs discovered using a state-of-the-art FD discovery algorithm [34] to govern the autoregressive data generation process.

FD Distillation. The result of traditional FD discovery, yields complex (i.e., multi-attribute) functional dependencies between columns which are ambiguous to resolve in an autoregressive generation setting. Hence we undertake an intermediate *FD distillation* step to simplify multi-attribute functional dependencies into multiple single attribute FDs as detailed below.



(a) There are two types of column dependency edges for three types of functional dependencies (FDs), which are distinguished by the left-hand side (LHS) and right-hand side (RHS) in the FD.



(b) DAG for column functional dependency derived by expanding SCC super nodes and retrieving a fully flattened, ordered structure.

Figure 2: FD-Distillation and Dependency Graph Sorting for automatically extracting order permutations from tables.

We first construct a dependency graph model $G(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ represents the set of vertices with each $v \in \mathcal{V}$ representing an attribute in $\mathcal{S}$ and $e_{ij} \in \mathcal{E}$ representing an edge relation from attribute v_i to attribute v_j. Further, we consider two types of edges in $\mathcal{E}$, specifically each e_{ij} may be a *type-1* edge or a *type-2* edge (defined next). The two edge types (i.e., *type-1*, *type-2*) in $\mathcal{E}$ are derived from three classes of FDs, as shown in Fig. 2a. Subsequently, we proceed to examine each individual case: 1) Single attribute left-hand side

(LHS) and single attribute right-hand side (RHS) 2) Multi-attribute LHS and single-attribute RHS. 3) single or multi-attribute LHS and multi-attribute RHS. We shall use the example table in Fig. 1 to define each FD case.

[**Type-1 Edge**]. Let us consider an example of FD Case 1, wherein the column *State* functionally determines column *Bird*. In such a case, we enforce that the value for the attribute *State* be generated prior to the value for the attribute *Bird*. Accordingly, a forward directed edge from *State* to *Bird* is created in the column dependency graph $\mathcal{G}$. We term such forward directed edges as type-1 edges in $\mathcal{G}$. [**Type-2 Edge**]. The other type of edge in $\mathcal{G}$, arises when we encounter an FD with a multi-attribute LHS and a single attribute RHS (i.e., FD Case 2). As per FD Case 2, the values of multiple columns in the LHS would collectively decide the value of the column on the RHS. As an example in Fig. 2a, the tuple of columns *Latitude, Longitude* functionally determines *State*. For such FDs, two backward edges are added in the column dependency graph $\mathcal{G}$, connecting *State* to both *Latitude* and *Longitude*. We term such backward directed edges as type-2 edges in $\mathcal{G}$.

FD Case 3 relationships are ones where the RHS has multiple attributes and the LHS could have single or multiple-attributes. Such relationships do not directly result in an edge in our dependency graph $\mathcal{G}$. Instead, as classically done in FD literature [34], we subject such FD relationships to an intermediate decomposition step. Specifically, the multi-attribute RHS of FD Case 3 relationships is decomposed into multiple single-attribute RHS dependencies each comprising the original LHS. Further, in each of these new decomposed relationships, if the LHS is single-attribute, it is treated as a FD *Case 1* relationship (i.e., a directed edge from *LHS* to *RHS* is added to $\mathcal{G}$), else it is handled as an FD *Case 2* relationship, wherein for each attribute in the multi-attribute LHS, a *backward* dependency edge from the single-attribute RHS is added to the dependency graph $\mathcal{G}$. Therefore, for every column in the right-hand side (RHS), we employ either *type-1* or *type-2* edge construction, depending on the value of its left-hand side (LHS). Full procedure detailed in Appendix B.1 Algorithm 1.

4.3 Putting It All Together

Until this point, our construction of the graph $\mathcal{G}$ has only been limited to considering pair-wise relationships between columns in $\mathcal{D}$. Graph properties like functional dependency transitivity, require us to obtain a total ordering on the nodes $v \in \mathcal{V}$ of $\mathcal{G}(\mathcal{V}, \mathcal{E})$ that is deterministic in nature for effective auto-regressive LLM training. This implies that in order to obtain a feature order permutation ($\mathbf{k}$) from the derived functional dependency relationships (Sec. 4.2), a computation must be performed on the entire dependency graph $G(\mathcal{V}, \mathcal{E})$.

We define this task of obtaining a total feature order $\mathbf{k}$ from $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as an optimization step which seeks to produce $\mathbf{k}$ while minimizing the number of violated relationships in $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Appendix B.1 Algorithm 2 outlines this procedure.

Consider the trivial case of having an empty FD graph $\mathcal{G}$ i.e., $|\mathcal{E}| = \emptyset$. If the columns of a table are not functionally dependent on each other, then the order of generation is not important and $\mathbf{k}$ can be some arbitrary permutation of the columns in $\mathcal{S}$. For all other cases, our total feature ordering algorithm operates in three phases, as shown in Fig. 2b. [**Phase 1: Condensation**]. It is apparent that if a graph is not a directed acyclic graph (DAG), there is no optimal solution to the total feature ordering problem. In other words, there must be FDs that cannot be satisfied in the resulting total order permutation $\mathbf{k}$. In such cases, we compute the strongly connected components (SCC), and condense them into super nodes, thus transforming the original graph into a DAG. [**Phase 2: Ordering**]. An application of a topological sort onto the DAG from Phase 1 will result in a total feature ordering with all SCCs in $\mathcal{G}$ compressed into super nodes. [**Phase 3: Expansion of SCC**]. Once the topological sort is conducted in Phase 2, we finally expand the SCC super nodes (via. arbitrary ordering) such that although the intra-SCC ordering of the nodes within the SCC is arbitrary, their ordering relative to non-SCC nodes is maintained.

4.4 Synthetic Data Generation using PAFT

After the optimized feature order permutation is obtained (via Appendix B.1 Algorithm 2), we fine-tune the LLM with the textually encoded table record $\mathbf{t}_i$ such that the auto-regressive generation process is governed by the optimal feature order permutation $\mathbf{k} = \phi(\mathcal{D}, S)$. Specifically, we generate the table governed by order $\mathbf{k}$ as defined in Eq. 4.

$$\mathbb{P}(\mathbf{t}_i) = \mathbb{P}(t_{i,k_1}, ..., t_{i,k_j}) \simeq \prod_{j=1}^{m} \mathbb{P}(t_{i,k_j} | t_{i,k_1}, .., t_{i,k_{j-1}}). \tag{4}$$

We employ the Low-Rank Adaptation (LoRA) fine-tuning strategy [19]. To generate synthetic rows, we first sample the initial token $p(t_{i,k_1})$ from the marginal distribution of variable k_1 in actual training data, and then use Eq. 4 to sequentially sample subsequent tokens $p(t_{i,k_j})$, where $j \in 2, ..., m$.

5 Experimental Evaluation

We conduct an exhaustive empirical evaluation of PAFT to assess its ability to reproduce realistic data distributions, superiority over other competing approaches, and most importantly substitutability of data generated by PAFT in the context of a larger ML pipeline. More specifically, the questions we seek to answer are:

(1) Does PAFT-generated synthetic data accurately capture conditional distributions within categories? (Sec 5.1)
(2) Does PAFT generate data respect the consistency of intrinsic data characteristics? (Sec 5.2)
(3) Does the synthetic data generated by PAFT pass the *sniff* test? (Sec 5.3)
(4) Can data generated by PAFT replace real data in downstream ML model training? (Sec 5.4)
(5) Do the data sets generated by PAFT adhere to real distributions and possess mode diversity? (Sec 5.5)
(6) Do newer generations of LLMs obviate the need for PAFT ? (Sec 5.6)

Datasets. We evaluate the effectiveness of PAFT through experiments on six real datasets commonly used in synthetic table generation studies such as GReaT [3] CTGAN [52]. These are Beijing [7], US-locations [14], California Housing [31], Adult Income [2], Seattle [43], and Travel [47]. Separately, we also generate a set of four simulated datasets for class-mixture distributions (as described in

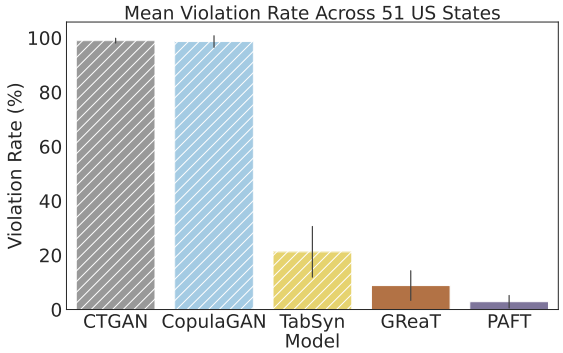


Figure 3: For a composite dataset, US-locations, this comparison examines state-specific violation rates across different synthetic data generation approaches. The error bars represent standard deviation. The states on the x-axis are ordered by decreasing violation rates. PAFT significantly reduces state-specific violations in the composite dataset.

Algorithm 3 Appendix C.1). Details of dataset statistics are in the Appendix C.1.

Baselines. For benchmarking, we organize baselines that utilize current deep learning approaches for synthetic data generation (CTGAN [52], CopulaGAN [52], TabSyn [55]) and the most advanced synthetic table generator with LLM fine-tuning GReaT [3]. To guarantee an equitable comparison, we employ the Distill-GReaT model for both LLM techniques in all tests.

Reproducibility. Each baseline (CTGAN, CopulaGAN, TabSyn, GReaT) adheres to the recommended hyperparameters and utilizes officially released API tools: Synthetic Data Vault [36] and GReaT [3]. For a fair comparison of GReaT and PAFT, the LoRA fine-tuning parameters are set the same as: Lora attention dimension $r = 16$, alpha parameter for Lora scaling $lora_alpha = 32$, the names of the modules to apply the adapter to $target_modules = c_attn$, the dropout probability for Lora layers $lora_dropout = 0.05$, $bias = none$.

Parameters for MLE and Discriminator Models. We utilize neural network, linear/logistic regression, and random forest models from the Scikit-Learn package for the ML efficiency and discriminator experiments. The exact hyperparameters for each model are detailed in Appendix Table 8. Every result is evaluated through the 5-fold cross-validation process.

Low-order statistics [55] of column-wise data distribution and pair-column correlation are calculated with the SDV library [2].

5.1 RQ1: Does PAFT-generated synthetic data accurately capture conditional distributions within categories?

Capturing and generating the diversity in a multi-class setting (composite dataset) has been shown to be a challenging practical problem [27]. In addition to Table 1, Fig. 3 and Table 2 further illustrate the widespread nature of this issue within the same composite dataset, where the GAN-generated results exhibit an almost 100% violation rate. The term 'composite dataset' refers to a dataset in which the distributions across different subcategories show significant variances. The LLM model GReaT, which employs random order permutation, shows a significant improvement in maintaining conditional distributions. Nevertheless, its performance remains inconsistent due to unevenness within categories, resulting in violation rates ranging from 0.5% to 22.3%. In contrast, PAFT consistently controls deviations from the real facts of conditional distributions, maintaining them within a range of 0% to 5%. This reliability holds even in challenging subcategory cases where baseline methods underperform.

[2]https://docs.sdv.dev/sdmetrics/reports/quality-report

Table 2: Violation rates across different categories in the same dataset highlight the effectiveness of PAFT in addressing conditional distribution challenges in mixed-category datasets. Notably, even though baseline models may perform well in the MLE task or distribution evaluation, they often fail in practical boundary checks, such as functional dependencies (FDs). We can see that states with the lowest violation rates are the easiest to model, i.e. rectangular.

States with the **Highest** Violation Rates (↓) (Sorted by LLM baseline: GReaT)					
Category (State)	MO	AK	KY	FL	WV
Real Data	0%	0%	0%	0%	0%
CTGAN	98.0%	99.3%	97.4%	99.9%	100%
CopulaGAN	99.3%	84.6%	99.3%	97.8%	100%
TabSyn	10.5%	17.2%	22.9%	25.1%	28.1%
GReaT	17.1%	18.4%	20.6%	21.9%	22.3%
PAFT	**1.9%**	**5.1%**	**3.4%**	**3.4%**	**3.4%**
States with the **Lowest** Violation Rates (↓) (Sorted by LLM baseline: GReaT)					
Category (State)	CO	KS	NM	SD	UT
Real Data	0%	0%	0%	0%	0%
CTGAN	96.5%	97.0%	99.5%	97.5%	99.3%
CopulaGAN	98.3%	98.8%	98.4%	98.3%	98.4%
TabSyn	11.3%	18.4%	6.2%	17.3%	16.6%
GReaT	0.5%	0.9%	1.2%	1.5%	2.1%
PAFT	**0.0%**	**0.3%**	**0.9%**	**0.9%**	**0.0%**

5.2 RQ2: Does PAFT generate data respecting the consistency of intrinsic data characteristics?

In addition to the dissimilar sub-category challenge, we also investigate whether unsatisfactory conditional distributions exist in

Table 3: Datasets have intrinsic characteristics like functional dependencies, range restrictions, and other domain knowledge. Results are averaged over five random runs, with variance detailed in Appx. Table 9.

Intrinsic Fact (Dataset)	Fact Violation Rate (↓)				
	CTGAN	Cop.GAN	TabSyn	GReaT	PAFT
Lat-long→State (US-locations)	99.2%	98.5%	21.5%	8.2%	**2.9%**
Lat-long → CA (California)	47.6%	99.9%	8.8%	5.4%	**1.3%**
Med. house price → $[1.4e^5, 5e^5]$ (California)	1.5%	0.01%	**0.0%**	**0.0%**	**0.0%**
education→ education-num (Adult)	83.9%	19.1%	1.4%	1.2%	**0.5%**
Zipcode→Seattle (Seattle)	**0.0%**	99.9%	**0.0%**	**0.0%**	**0.0%**

general across various datasets, and whether the PAFT method can address these issues. In line with this, we conducted rule checks that were derived from real-world scenarios and subsequently evaluated the generated data from all models. Table 3 displays the violation rate in the generated data. From the table, we observe that PAFT adheres to data's characteristics more faithfully (i.e., significantly fewer rule violations) than baseline methods, by learning together with functional dependencies.

5.3 RQ3: Does the synthetic data generated by PAFT pass the *sniff* test?

Similarly to the analysis conducted in recent work [3] (GReaT), we employ the random forest (RF) algorithm to train discriminators to distinguish real data (labeled True) and synthetically generated data (labeled False). Subsequently, we test performance on an unseen set (consisting of 50% synthetically generated data and 50% real data). In this experiment, scores represent the percentage of correctly classified entities. In this case, an ideal accuracy score would be close to 50%, which means the discrimniator fails to distinguish between real and synthesized data. The scores are shown in Table 4 and indicate that the data generated by PAFT are most indistinguishable from real data, even by powerful discriminative models.

5.4 RQ4: Can data generated by PAFT replace real data in downstream ML model training?

We next assess the effectiveness of the generated (synthetic) data by comparing the performance of discriminative models trained on synthetic data versus real data for their target tasks. Models tested include random forests (RF), linear regression (LR), and multi-layer perceptron (NN). As shown in Table 5, PAFT is best or second best in over 80% of (dataset, method) combinations.

Table 4: Privacy Discriminator Performance. The scores stand for the accuracy for detecting real or fake data, where the ML models are trained using 50% real data and 50% random data. An ideal accuracy score is 50, indicating the model cannot distinguish between real and synthesized data. The best results are marked in bold, the second-best results are underlined. Results are averaged over five random runs, with variance detailed in Appx. Table 13.

Data Sniff Test - ML Discriminator Accuracy (Values closest to 50% are best.)					
Method	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
Beijing	99.16%	98.69%	50.97%	51.1%	**50.09%**
US-locations	99.94%	97.74%	51.97%	50.47%	**50.01%**
California	98.35%	86.64%	50.64%	53.74%	**49.89%**
Adult	94.43%	59.82%	51.64%	**51.12%**	48.75%
Seattle	87.61%	85.7%	**50.12%**	68.27%	47.21%
Travel	77.96%	74.14%	**50.66%**	62.49%	48.18%

Table 5: MLE performance. MLE performance: For datasets with regression tasks (marked *), performance is evaluated using MAPE (where lower scores are better). For datasets with classification tasks, accuracy is used (where higher scores are better). The best results are marked in bold and the second-best results are underlined. Results are averaged over five random runs, with variance in Appx. Table 10.

Regression		**MAPE (↓)** Over Different Methods					
Dataset(*)		Orig.	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
Beijing	**RF**	0.41%	2.49%	2.15%	0.7%	0.57%	**0.52%**
	LR	1.37%	2.23%	1.55%	1.25%	**0.97%**	1.34%
	NN	0.99%	2.44%	2.83%	1.01%	1.16%	**0.95%**
Calif.	**RF**	0.18%	0.65%	0.39%	0.22%	0.25%	**0.20%**
	LR	0.30%	0.54%	0.5%	0.30%	**0.29%**	0.31%
	NN	0.34%	0.53%	0.47%	0.29%	0.3%	**0.27%**
Seattle	**RF**	0.33%	0.76%	0.38%	0.30%	0.35%	**0.28%**
	LR	0.29%	0.74%	0.32%	**0.23%**	0.33%	0.29%
	NN	0.28%	0.71%	0.38%	0.28%	0.33%	**0.27%**
Classif.		**Accuracy (↑)** Over Different Methods					
Dataset		Orig.	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
US-loc.	**RF**	99.95%	7.17%	45.33%	**99.99%**	99.84%	99.91%
	LR	46.1%	5.11%	31.08%	43.69%	45.65%	**49.41%**
	NN	99.85%	7.56%	53.34%	**99.64%**	98.94%	99.44%
Adult	**RF**	84.97%	71.15%	81.33%	83.69%	**83.89%**	83.06%
	LR	78.53%	75.68%	78.18%	**78.38%**	76.1%	77.24%
	NN	76.9%	75.69%	76.6%	78.36%	78.23%	**79.16%**
Travel	**RF**	88.95%	56.35%	67.18%	84.09%	79.78%	**85.19%**
	LR	82.87%	70.17%	79.56%	**83.31%**	78.34%	82.76%
	NN	81.77%	71.05%	79.56%	81.88%	80.77%	**83.20%**

5.5 RQ5: Do the data sets generated by PAFT adhere to real distributions and possess mode diversity?

We also evaluate how closely the density and diversity of the true data distribution are matched by PAFT generated data using correlation metrics and density-based distance metrics. Specifically, we employ the Kolmogorov-Smirnov Test (KST) to evaluate the density estimate of numerical columns, and the Total Variation Distance (TVD) for categorical columns. When calculating the correlation between columns, we employ Pearson correlation for numerical columns and contingency similarity for categorical columns. The results for both density estimate similarity and correlation based analysis are detailed in Table 6. As shown, PAFT synthetic data closely matches the real data in terms of univariate distribution and bivariate correlation, outperforming the baseline.

Table 6: Low-order statistics [55] of column-wise data density and pair-wise column correlation[2]. Scores range from 0 to 1. Higher values indicate more accurate estimation. PAFT outperforms the best generative baseline model in most case. The best results are marked in bold, the second-best results are underlined. Results are averaged over five random runs, with variance in Appx. Table 11 and 12.

Single-Column Shape Score (↑)					
Dataset	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
Adult	0.81	0.92	**0.98**	0.88	0.90
Beijing	0.89	0.79	**0.98**	0.93	0.97
California	0.87	0.77	**0.98**	0.89	0.83
US-locations	0.83	0.82	0.96	0.93	**0.97**
Seattle	0.83	0.73	0.93	0.90	**0.93**
Travel	0.84	0.90	0.93	**0.93**	**0.93**
Two-Column Pair Trends score (↑)					
Dataset	CTGAN	Co.GAN	TabSyn	GReaT	**PAFT**
Adult	0.81	0.86	**0.93**	0.80	0.78
Beijing	0.92	0.94	**0.99**	0.95	0.98
California	0.84	0.87	**0.97**	0.87	0.91
US-locations	0.50	0.55	0.93	0.89	**0.94**
Seattle	0.74	0.72	0.80	0.76	**0.81**
Travel	0.77	0.80	**0.87**	0.85	0.82

Visual examples are depicted in Figure 4. PAFT has the ability to generate a wide range of diversity, encompassing both continuous and discrete variables, which closely resembles real data.

5.6 RQ6: Do newer generations of LLMs obviate the need for PAFT ?

The choice of foundation models for synthetic table generation involves two key considerations: in-context learning and fine-tuning,

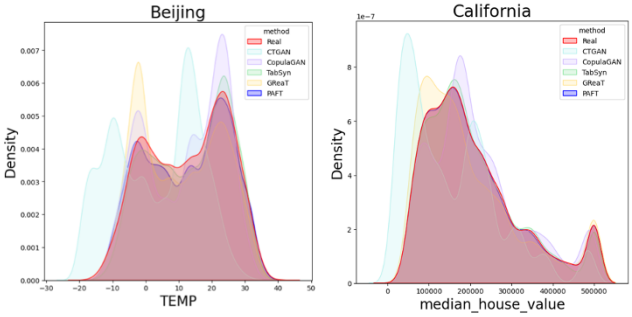


Figure 4: As a supplementary illustration of Table 6 we provide column distribution visualizations. Overall, PAFT (Blue) most closely matches the real data distribution (Red) compared to other synthesis methods, while also demonstrating valid and diverse outputs. Extended visualizations are available in Appendix Figure 7.

especially with large-parameter models such as GPT-4 and LLaMA. A recent survey [15] indicates that the in-context learning approach of GPT-4 is well-suited for augmenting tabular data in low-data regimes, as highlighted by CLLM [44]. However, GPT-4 suffers from limitations such as the disappearance of column-wise tail distributions and a low success rate in accurately extracting output cell values [15].

In contrast, the state-of-the-art GReaT [3] and other synthetic data generation approaches [46, 56, 58] typically employ smaller models like GPT-2 or DistilGPT2, which effectively address attribute encoding while reducing feature engineering efforts. These smaller models already outperform traditional GAN and traditional statistics-based approaches. For example, DistilGPT2 can be trained using LoRA on GPUs as modest as the Tesla P100.

Importantly, fine-tuning larger models such as GPT-4 entails significant computational cost. For instance, fine-tuning a GPT-4 model for three epochs on a dataset with 50k rows and a five-column table costs approximately $300 (OpenAI) or requires equivalent high-end GPU resources. Therefore, while the same fine-tuning schema can be applied to models like GPT-4 or LLaMA, it is not a cost-effective solution compared to the solutions considered here.

In summary, newer generation LLMs do not obviate the need for PAFT . Our method directly addresses the impedance mismatch between autoregressive LLMs and synthetic table generation—particularly by preserving functional dependencies through permutation-aware fine-tuning—a challenge that persists even with advanced models.

6 Conclusion

This work has brought LLMs closer to the goal of generating realistic synthetic datasets. By learning FDs and leveraging this information in the fine-tuning process, we are able to align the autoregressive nature of LLMs with the ordering of columns necessary for generating quality synthetic data. While PAFT is quite broadly applicable by itself, it can be extended in several directions. First, what are other, perhaps more expressive, types of tabular constraints that can be utilized in the fine-tuning process? Second,

what is the internal basis for regulating orders inside a transformer architecture and can we more directly harness it? Third, can we theoretically prove the (im)possibility of generating specific synthetic datasets by LLM architectures? Fourth, despite the numerous advantages of LLM in learning and generating tabular data, scalability remains an acknowledged challenge [3, 10, 15], encompassing concerns such as context window and training speed. And finally, privacy-preserving methods have been implemented in table generators based on GANs but remains understudied in LLM fine-tuning. These questions will be the focus of our future work.

Limitations. The row-wise generation cost of our method, particularly when employing fine-tuning, is affected by the dataset sample size and computational resources (GPU). Moreover, the capacity of PAFT to generate columns is affected by context window sizes. These limitations can be overcome by the newer generation of LLMs or by exploring partial row generation, i.e., generating a row in multiple steps using an LLM.

Acknowledgments

This work is supported in part by US National Science Foundation grant IIS-2312794. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Ronny Kohavi Barry Becker. 1996. Adult. https://archive.ics.uci.edu/dataset/2/adult

[3] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. 2022. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280* (2022).

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and VS Subrahmanian. 2019. FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data.. In *IJCAI*. 2074–2080.

[6] Hongjie Chen, Ryan A Rossi, Kanak Mahadik, Sungchul Kim, and Hoda Eldardiry. 2023. Graph Deep Factors for Probabilistic Time-series Forecasting. *ACM Transactions on Knowledge Discovery from Data* 17, 2 (2023), 1–30.

[7] Song Chen. 2017. Beijing PM2.5 Data. https://archive.ics.uci.edu/dataset/381/beijing+pm2+5+data

[8] Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise Order Matters in Reasoning with Large Language Models. *arXiv preprint arXiv:2402.08939* (2024).

[9] Milan Cvitkovic. 2020. Supervised learning on relational databases with graph neural networks. *arXiv preprint arXiv:2002.02046* (2020).

[10] Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy-yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. 2022. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. *Advances in Neural Information Processing Systems* 35 (2022), 11763–11784.

[11] Bayu Distiawan, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1627–1637.

[12] Lun Du, Fei Gao, Xu Chen, Ran Jia, Junshan Wang, Jiang Zhang, Shi Han, and Dongmei Zhang. 2021. TabularNet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 322–331.

[13] Yuntao Du and Ninghui Li. 2024. Towards principled assessment of tabular data synthesis algorithms. *arXiv preprint arXiv:2402.06806* (2024).

[14] Dominique Evans-Bye. 2015. States Shapefile. https://hub.arcgis.com/datasets/CMHS::states-shapefile/explore?location=29.721532%2C71.941464%2C3.76.

[15] Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun (Jane) Qi, Scott Nickleach, Diego Socolinsky, "SHS" Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models (LLMs) on tabular data: Prediction, generation, and understanding — a survey. *Transactions on Machine Learning Research* (2024).

[16] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. 2008. *Database Systems: The Complete Book* (2 ed.). Prentice Hall Press, USA.

[17] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[18] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. *CoRR* abs/2203.15556 (2022). doi:10.48550/ARXIV.2203.15556 arXiv:2203.15556

[19] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *CoRR* abs/2106.09685 (2021). arXiv:2106.09685 https://arxiv.org/abs/2106.09685

[20] James Max Kanter and Kalyan Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 1–10.

[21] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[22] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110* (2022).

[23] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2020. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In *Proceedings of the ACM Internet Measurement Conference*. 464–483.

[24] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).

[25] Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. 2022. GOGGLE: Generative modelling for tabular data by learning relational structure. In *The Eleventh International Conference on Learning Representations*.

[26] Panagiotis Mandros, David Kaltenpoth, Mario Boley, and Jilles Vreeken. 2020. Discovering functional dependencies from mixed-type data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1404–1414.

[27] Andrei Margeloiu, Xiangjian Jiang, Nikola Simidjievski, and Mateja Jamnik. [n. d.]. TabEBM: A Tabular Data Augmentation Method with Distinct Class-Specific Energy-Based Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[28] 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.

[29] R. E. Miller and P. D. Blair. 2009. *Input-output analysis: foundations and extensions*. Cambridge university press.

[30] Nikhil Muralidhar, Chen Wang, Nathan Self, Marjan Momtazpour, Kiyoshi Nakayama, Ratnesh Sharma, and Naren Ramakrishnan. 2018. illiad: Intelligent invariant and anomaly detection in cyber-physical systems. *ACM Transactions on Intelligent Systems and Technology (TIST)* 9, 3 (2018), 1–20.

[31] Cam Nugent. 2017. California Housing Prices. https://www.kaggle.com/datasets/camnugent/california-housing-prices

[32] OpenAI. 2022. Chat-GPT: Optimizing Language Models for Dialogue. https://openai.com/blog/chatgpt/

[33] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment* 8, 10 (2015), 1082–1093.

[34] Thorsten Papenbrock and Felix Naumann. 2016. A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 International Conference on Management of Data*. 821–833.

[35] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1071–1083.

[36] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. 2016. The Synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 399–410. doi:10.1109/DSAA.2016.49

[37] Frédéric Pennerath, Panagiotis Mandros, and Jilles Vreeken. 2020. Discovering approximate functional dependencies using smoothed mutual information. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge*

Discovery & Data Mining. 1254–1264.

[38] Ben Prystawski, Michael Li, and Noah Goodman. 2024. Why think step by step? Reasoning emerges from the locality of experience. *Advances in Neural Information Processing Systems* 36 (2024).

[39] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[40] C. Raffel, N. Shazeer, A. Roberts, K. Lee, Sharan S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[41] Leonardo FR Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-Text Generation with Dual Graph Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3183–3194.

[42] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. 2023. Mathematical discoveries from program search with large language models. *Nature* (2023), 1–3.

[43] samuel Cortinhas. 2022. HOUSE PRICE PREDICTION - SEATTLE. https://www.kaggle.com/datasets/samuelcortinhas/house-price-prediction-seattle

[44] Nabeel Seedat, Nicolas Huynh, Boris van Breugel, and Mihaela van der Schaar. 2024. Curated LLM: Synergy of LLMs and Data Curation for tabular augmentation in low-data regimes. In *Forty-first International Conference on Machine Learning*.

[45] Mandar Sharma, Ajay Kumar Gogineni, and Naren Ramakrishnan. 2024. Neural Methods for Data-to-text Generation. *ACM Transactions on Intelligent Systems and Technology* (2024).

[46] Aivin V Solatorio and Olivier Dupriez. 2023. Realtabformer: Generating realistic relational and tabular data using transformers. *arXiv preprint arXiv:2302.02041* (2023).

[47] Tejashvi. 2021. Tour & Travels Customer Churn Prediction. https://www.kaggle.com/datasets/tejashvi14/tour-travels-customer-churn-prediction

[48] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[50] Yun Wang, Zhida Sun, Haidong Zhang, Weiwei Cui, Ke Xu, Xiaojuan Ma, and Dongmei Zhang. 2020. DataShot: Automatic Generation of Fact Sheets from Tabular Data. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 895–905. doi:10.1109/TVCG.2019.2934398

[51] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1780–1790.

[52] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional gan. *Advances in neural information processing systems* 32 (2019).

[53] Lei Xu and Kalyan Veeramachaneni. 2018. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264* (2018).

[54] Shengzhe Xu, Manish Marwah, Martin Arlitt, and Naren Ramakrishnan. 2021. Stan: Synthetic network traffic generation with generative neural models. In *Deployable Machine Learning for Security Defense: Second International Workshop, MLHat 2021, Virtual Event, August 15, 2021, Proceedings 2*. Springer, 3–29.

[55] Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. 2023. Mixed-Type Tabular Data Synthesis with Score-based Diffusion in Latent Space. *arXiv preprint arXiv:2310.09656* (2023).

[56] Tianping Zhang, Shaowen Wang, Shuicheng Yan, Li Jian, and Qian Liu. 2023. Generative Table Pre-training Empowers Models for Tabular Prediction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 14836–14854.

[57] Yunjia Zhang, Zhihan Guo, and Theodoros Rekatsinas. 2020. A statistical perspective on discovering functional dependencies in noisy data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 861–876.

[58] Zilong Zhao, Robert Birke, and Lydia Chen. 2023. Tabula: Harnessing language models for tabular data synthesis. *arXiv preprint arXiv:2310.12746* (2023).

[59] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. 2021. Ctab-gan: Effective table data synthesizing. In *Asian Conference on Machine Learning*. PMLR, 97–112.

[60] Lei Zheng, Ning Li, Xianyu Chen, Quan Gan, and Weinan Zhang. 2023. Dense Representation Learning and Retrieval for Tabular Data Prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3559–3569.

[61] Yujin Zhu, Zilong Zhao, Robert Birke, and Lydia Y Chen. 2022. Permutation-Invariant Tabular Data Synthesis. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 5855–5864.

A Related Work

Owing to the ubiquity of tabular data, the synthetic generation of this type of data in traditional machine learning research. Various approaches have been developed for tabular data generation.

Tabular Data Generation with Neural Networks (i.i.d. rows). Lei et al. [52] proposed CTGAN where rows are independent of each other; a conditional GAN architecture ensures that the dependency between columns is learned. Tabsyn [55] also generates independent rows but with a diffusion approach.

Tabular Data Generation with Neural Networks (non i.i.d. rows). DoppelGanger [23] uses a combination of an RNN and a GAN to incorporate temporal dependencies across rows but this method has been tested in traditional, low-volume settings such as Wikipedia daily visit counts. For high-volume applications, STAN [54] utilizes a combination of a CNN and Gaussian mixture neural networks to generate synthetic network traffic data. GraphDF [6] conducts multi-dimensional time series forecasting. GOGGLE [25] employs a generative modeling method for tabular data by learning relational structures.

Use of Language Models (LLMs) for tabular data generation. Most modern LLMs are based on the transformer architecture [49] with parameters ranging from few millions to billions [18], and researchers have developed creative ways to harness LLMs in traditional machine learning and data contexts. LIFT [10] initially transforms a table row into a sentence, such as 'An Iris plant with sepal length 5.1cm, sepal width 3.5cm', and employs an LLM as a learning model for table classification, regression, and generation tasks. GReaT [3], introduced earlier, utilizes a GPT-2 model that has been fine-tuned using a specific corpus for synthetic data generation. A general benefit of utilizing LLMs is the elimination of customized preprocessing pipelines.

Feature Ordering. Although not well-studied in the context of tabular data generation, the notion of feature ordering has been investigated in the context of graph-to-text translation [45] wherein to learn effective graph encodings, vertices are linearized via combinations of different graph traversal mechanisms, e.g., topological & breath-first strategies [11], and top-down & bottom-up approaches [41]. As a second example, 'permutation-invariant tabular data synthesis [61] examines the influence of the arrangement of table columns on convolutional neural network (CNN) training and organizes them according to the correlation among columns. Nevertheless, it is important to acknowledge that relying on mere correlation to establish column orders can be limiting. There are also several approaches (e.g., [9, 20]) that synthesize, discover, or aggregate features from relational databases, leveraging order information when possible, for use in machine learning pipelines. It worth to note that even in the LLM community, the task of context sorting for LLM prompting is not trivial and has gained significant attention lately [8].

Mining and Modeling Functional Dependencies. Yunjia et al. [57] relax the notion of strict functional dependencies to include noisy functional relationships by utilizing probabilistic graphical models. Chen et al. [5], in their FakeTables approach use the discovery of functional dependencies in a GAN formulation; they first use a generator to create a set of columns (set A) and an autoencoder to cast another set (set B), which are then used by a discriminator to calculate the gradient loss. Muralidhar et al. [30] proposed to use Granger causality to incorporate functional *invariants* across multiple time series. The area of FD discovery and of data mining with tabular data have both been extensively studied [26, 37, 60].

B Extended Methodology

B.1 Extended Algorithm

Algorithm 1 details FD distillation procedures corresponding to the text in Section 4.2 and Algorithm 2 details Feature Order Permutation Optimization procedures corresponding to the text in Section 4.3.

Algorithm 1 FD Distillation with Schema-Level FDs

Require: List of Schema-level FDs, $\mathcal{S}$
Ensure: Column Dependency Graph, $\mathcal{G}(\mathcal{V}, \mathcal{E})$

```
G ← ∅
for fd ∈ S do
    LHS, RHS ← fd
    if LHS.length = 1 and RHS.length = 1 then
        for u ∈ LHS do
            for v ∈ RHS do
                G.add_edge(u, v)                    ▷ Case 1
    else if LHS.length > 1 and RHS.length = 1 then
        for u ∈ LHS do
            G.add_edge(v, u)                        ▷ Case 2
    else                                            ▷ RHS.length > 1, Case 3
        for v ∈ RHS do
            if LHS.length = 1 then
                G.add_edge(u, v)                    ▷ Go to Case 1
            else
                for u ∈ LHS do
                    G.add_edge(v, u)                ▷ Go to Case 2
```

Algorithm 2 Feature Order Permutation Optimization

Require: Column Dependency Graphs $\mathcal{G}(\mathcal{V}, \mathcal{E})$
Ensure: Optimal Feature Order Permutation $\mathbf{k}$

```
if |E| = ∅ then
    Return k ← arbitrary_permutation(V)
if G is not a Directed Acyclic Graph (DAG) then     ▷ Phase 1
    G ← strongly_connected_components(G)
k ← topological_ordering(G)                         ▷ Phase 2
k ← arbitrary_SCC_expansion(G)                      ▷ Phase 3
Return k
```

C Data and Experiment Description

C.1 Experimental Setup

Dataset. We evaluated the efficiency of PAFT through experiments on six real datasets commonly used in synthetic table generation studies (GReaT, CTGAN, etc.), as well as a set of four simulated datasets: Beijing [7], US-locations [14], California Housing [31], Adult Income [2], Seattle [43], and Travel [47], Simulated (Algorithm 3). These real-world datasets come from diverse domains

and vary in size. The range of the number of functional dependencies spans from 0, indicating *complete independence* between columns, to around 400, indicating a high level of *interdependence*. These data also demonstrate the diverse combinations of categories and numerical columns. The simulated data is customized to adhere to the given functional dependence schema, thus explicitly emphasizing the degree to which a model can precisely represent the functional relationship. The simulated data has four distinct versions, denoted by the variable k, which represents the unique values in the d column. As the value of k increases, the data becomes more complex, which has been demonstrated to make training the generative model more challenging, as evidenced by all the experimental results presented in this paper. In particular, the functional dependency graph of the simulated data is [$a \rightarrow b$, $a \rightarrow c$,$b \rightarrow c$,$b \rightarrow d$,$a \rightarrow d$,$b \rightarrow d$]. Table 7 provides the FD characteristics of each dataset, while Table 7 provides an more detailed overview.

Training and Testing. To prevent any data leakage, we partitioned the data sets into 80% training sets and 20% test sets. All models are trained or fine-tuned on the same training data samples. All models undergo cross-validation using 5 generated data sets to validate their results. One advantage of using LLM for creating tabular data is that there is no need for any complex data preparation. This means that the feature names and values are used just as they are supplied in the original data sets.

Dataset	#Rows	#Cat	#Num	**#FD**
Beijing	43,824	1	12	**157**
US-locations	20,400	3	2	**7**
California	20,640	8	0	**362**
Adult	32,561	9	6	**78**
Seattle	2,016	2	6	**10**
Travel	954	4	3	**0**
Simulated,k=[1,5,10,15]	10,000	4	0	**6**

Table 7: Dataset Descriptors (number of rows, categorical columns, numerical columns, and FDs).

Algorithm 3 Building simulated data: Given a dependency graph G, setting values for a table with n rows and m columns with different statistic complexity based on the initial unique value k for the complexity in the root column of the functional dependency chain.

```
procedure SetValues(n, m, G, k)
    table[n][m] ← {}
    top_order, node_layer ← Algorithm 2(G)
    unique_value ← {}
    for vertex ∈ topo_order do
        top_value[vertex] = 2^(max_layer−node_layer[vertex])
    for j ← 1 to m do
        for i ← 1 to n do
            table[i][j] ← x_(i%(top_value[j]*k))
```

Baselines. In benchmarking suite, we have baselines that consist of current deep learning approaches for synthetic data generation (CTGAN [52], CopulaGAN [52]), and the most advanced LLM fine-tuning synthetic table generator GReaT [3]. To guarantee an equitable comparison, we employ the Distill-GReaT model for both techniques in all tests, and adjust the hyperparameters as advised by the official GitHub website of GReaT. It ought to mention that GReaT utilizes textual encodings with random feature order permutations. This implies that each sample will undergo a different random order during the training and sampling process. This strategy appears similar to the edge case in Algorithm 2, but in fact, they are distinct. When the FD set is empty, PAFT will suggest a random permutation. Nevertheless, this permutation serves as a comprehensive guide for all samples after an in-depth assessment of FD.

C.2 Reproducibility detail

Baselines. Each baseline (CTGAN, CopulaGAN, TabSyn, GReaT) sticks to the recommended hyperparameters and utilizes officially released API tools: Synthetic Data Vault [36] and GReaT [3]. As for the fair comparison of GReaT and PAFT, the LoRA fine-tuning parameters are set as: Lora attention dimension $r = 16$, alpha parameter for Lora scaling $lora_alpha = 32$, The names of the modules to apply the adapter to $target_modules = c_attn$, The dropout probability for Lora layers $lora_dropout = 0.05$, $bias = none$.

Computational Resources. To ensure fairness in the comparison between the baselines, all baseline models and experiments were executed on a single Tesla P100-PCIE-16GB GPU.

Parameters for MLE and Discriminator Models. We utilize neural network, linear/logistic regression, and random forest models from the Scikit-Learn package for the ML efficiency and discriminator experiments. The exact hyperparameters for each model are detailed in Table 8. Every result is evaluated through the process of 5-fold cross-validation.

Low-order statistics [55] of column-wise data density is calculated with SDV library.

D Additional Research Questions and Case Studies

D.1 RQ2: Does PAFT generate data respecting the consistency of intrinsic data characteristics?

Table 9 details the standard deviations from five experimental runs.

D.2 RQ3: Can data generated by PAFT replace real data in downstream ML model training?

Table 10 details the standard deviations from five experimental runs.

	RF		LR	NN		
	n_est	max_depth	max_iter	max_iter	hidden_layer_sizes	learning_rate
Classification	100	*None*	100	300	(150, 100, 50)	0.001
Regression	100	*None*	100	300	(150, 100, 50)	0.001

Table 8: The parameters we used for MLE and Discriminator models remain the same across all datasets.

Dataset	Intrinsic Fact	Fact Violation Rate (↓)				
		CTGAN	CopulaGAN	TabSyn	GReaT	PAFT
US-locations	State-code → Bird	94.66±0.32%	95.66±0.17%	0.10±0.04%	0.30±0.00%	**0.00±0.00%**
US-locations	Lat-long → State	99.22±0.08%	98.51±0.07%	21.50±0.32%	8.16±0.14%	**2.93±0.15%**
California	Lat-long → CA	47.56±6.37%	99.93±0.00%	8.83±0.13%	5.42±0.16%	**1.26±0.06%**
California	Median house price → $[1.4e^5, 5e^5]$	1.46±1.52%	0.01±0.01%	**0.00±0.00%**	**0.00±0.00%**	**0.00±0.00%**
Adult	education → education-num	83.94± 1.93%	19.09± 0.58%	1.43±0.04%	1.24±0.09%	**0.46± 0.03%**
Seattle	Zipcode → Seattle	**0.00±0.00%**	99.88±0.00%	**0.00±0.00%**	**0.00±0.00%**	**0.00±0.00%**

Table 9: Additional Std. details in intrinsic characteristics evaluation.

D.3 RQ4: Does PAFT adhere to real distribution and possess mode diversity?

Tables 11 demonstrate that the PAFT synthetic data closely matches the real data in terms of univariate distribution and bivariate correlation, outperforming the baseline. Visual examples are depicted in Figure 7. PAFT has the ability to generate a wide range of diversity, encompassing both continuous and discrete variables, which closely resembles real data.

D.4 RQ5: Does the synthetic data generated by PAFT pass the *privacy* test?

Similar to the analysis conducted in recent work [3] (GReaT), we employ the random forest (RF) algorithm to train discriminators

Figure 5: Semantic Complexity: using the random order permutation to modeling the mixture of states data is more challenging when the rectangularity index (left) and compactness index (right) increase.

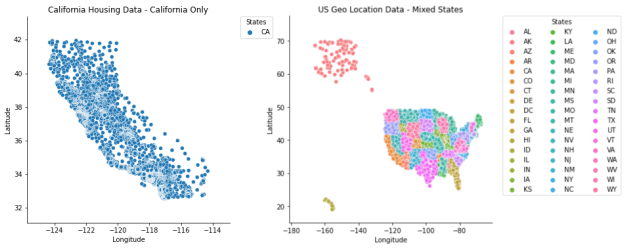


Figure 6: Statistical Complexity can be figured out by analyzing the distribution of the data. For instance, California Housing data is simpler to model concerning the Functional Dependency as it only includes the longitude and latitude for a single state.

to distinguish real data (labelled as True) and synthetically generated data (labeled as False). Subsequently, we test performance on an unseen set (consisting of 50% synthetically generated data and 50% real data). In this experiment, scores represent the percentage of correctly classified entities. In this case, an ideal accuracy score would be close to 50%, which means the discrimniator fails to distinguish between real and synthesized data. Scores are shown in Table 13 and indicate that the data generated by PAFT is most indistinguishable from real data, even by powerful discriminative models.

Regression Task		MAPE (↓) Over Different Methods					
Dataset		**Orig.**	**CTGAN**	**CopulaGAN**	**TabSyn**	**GReaT**	**PAFT**
Beijing (*)	**RF**	0.41%	2.49±0.57%	2.15±0.29%	0.7±0.01%	0.57±0.00%	**0.52±0.00%**
	LR	1.37%	2.23±0.54%	1.55±0.21%	1.25±0.0%	**0.97±0.01%**	1.34±0.01%
	NN	0.99%	2.44±0.74%	2.83±1.18%	1.01±0.14%	1.16±0.56%	**0.95±0.13%**
California (*)	**RF**	0.18%	0.65±0.09%	0.39±0.01%	0.22±0.0%	0.25±0.00%	**0.20±0.00%**
	LR	0.30%	0.54±0.1%	0.5±0.01%	0.30±0.0%	**0.29±0.00%**	0.31±0.00%
	NN	0.34%	0.53±0.11%	0.47±0.02%	0.29±0.02%	0.3±0.01%	**0.27±0.00%**
Seattle (*)	**RF**	0.33%	0.76±0.34%	0.38±0.07%	0.30±0.06%	0.35±0.01%	**0.28±0.03%**
	LR	0.29%	0.74±0.35%	0.32±0.03%	**0.23±0.04%**	0.33±0.00%	0.29±0.03%
	NN	0.28%	0.71±0.33%	0.38±0.08%	0.28±0.01%	0.33±0.00%	**0.27±0.01%**
Classification Task		**Accuracy (↑) Over Different Methods**					
Dataset		**Orig.**	**CTGAN**	**CopulaGAN**	**TabSyn**	**GReaT**	**PAFT**
US-locations	**RF**	99.95%	7.17±1.58%	45.33±2.82%	**99.99±0.01%**	99.84±0.07%	99.91±0.03%
	LR	46.1%	5.11±3.14%	31.08±1.92%	43.69±1.9%	45.65±0.86%	**49.41±1.57%**
	NN	99.85%	7.56±4.61%	53.34±1.59%	**99.64±0.17%**	98.94±1.16%	99.44±0.28%
Adult	**RF**	84.97%	71.15±5.59%	81.33±1.53%	83.69±0.28%	**83.89±0.42%**	83.06±0.35%
	LR	78.53%	75.68±0.14%	78.18± 1.53%	**78.38±0.11%**	76.1±0.29%	77.24±0.09%
	NN	76.9%	75.69±0.10%	76.6±1.26%	78.36±1.23%	78.23±1.31%	**79.16±0.15%**
Travel	**RF**	88.95%	56.35±2.64%	67.18±3.0%	84.09±1.18%	79.78±1.29%	**85.19±1.99%**
	LR	82.87%	70.17±17.46%	79.56±0.00%	**83.31±0.22%**	78.34±2.02%	82.76±1.01%
	NN	81.77%	71.05±17.85%	79.56±0.00%	81.88±1.33%	80.77±1.33%	**83.2±0.90%**

Table 10: MLE Performance (%): Comparison of original data to synthetic data.For datasets denoted as (*), we use a regression model for prediction, and calculate MAPE as performance (where lower scores are ideal); For other datasets, classification models are used for prediction and we calculate the accuracy as performance. The best results are marked in bold and the second-best results are underlined. RF: random forests; LR: linear regression: NN: a traditional multi-layer perceptron.

Dataset	CTGAN	CopulaGAN	TabSyn	GReaT	PAFT
Adult	0.81±0.01	0.92±0.01	**0.98±0.00**	0.88±0.00	0.90±0.00
Beijing	0.89±0.01	0.79±0.01	**0.98±0.00**	0.93±0.00	0.97±0.00
California	0.87±0.03	0.77±0.01	**0.98±0.00**	0.89±0.00	0.83±0.00
US-locations	0.83±0.02	0.82±0.00	0.96±0.00	0.93±0.00	**0.97±0.00**
Seattle	0.83±0.01	0.73±0.02	0.93±0.00	0.90±0.00	**0.93±0.00**
Travel	0.84±0.01	0.90±0.02	0.93±0.01	**0.93±0.01**	**0.93±0.01**

Table 11: Error rate (%) of column-wise density estimation[2]. Bold Face represents the best score on each dataset. Higher values indicate more accurate estimation (superior results). PAFT outperforms the best generative baseline model in most case. The best results are marked in bold, the second-best results are underlined.

D.5 RQ6: Can PAFT enhance the stability in generating high quality tables, resulting in a faster sampling phase?

When it comes to the comparison between fine-tuning approaches in LLM, the quality of the generated table rows can be reflected in the sampling process. This is particularly evident when generating an i.i.d row that involves auto-regressive generation. If the generated row cannot be decoded back into a real table row, then another sample needs to be redone. Given that the device condition and hyperparameters of the GReaT and PAFT are identical, a shorter sampling time indicates a higher probability of accepting a generated row, meaning improved generation quality. (See Table 14.) Furthermore, the time measurement is based solely on a single-core GPU to ensure a fair comparison of different baselines. Using multiple GPU parallel computing can significantly speed up the fine-tuning process in practice.

Dataset	CTGAN	CopulaGAN	TabSyn	GReaT	PAFT
Adult	0.81±0.02	0.86±0.01	**0.93±0.00**	0.80±0.01	0.78±0.00
Beijing	0.92±0.01	0.94±0.01	**0.99±0.00**	0.95±0.00	0.98±0.01
California	0.84±0.00	0.87±0.01	**0.97±0.00**	0.87±0.01	0.91±0.02
US-locations	0.50±0.02	0.55±0.00	0.93±0.00	0.89±0.00	**0.94±0.00**
Seattle	0.74±0.02	0.72±0.01	0.80±0.01	0.76±0.03	**0.81±0.01**
Travel	0.77±0.02	0.80±0.02	**0.87±0.01**	0.85±0.01	0.82±0.05

Table 12: Error rate (%) of pair-wise column correlation score[2]. Bold Face represents the best score on each dataset. PAFT outperforms the best baseline model in most case. The best results are marked in bold, the second-best results are underlined.

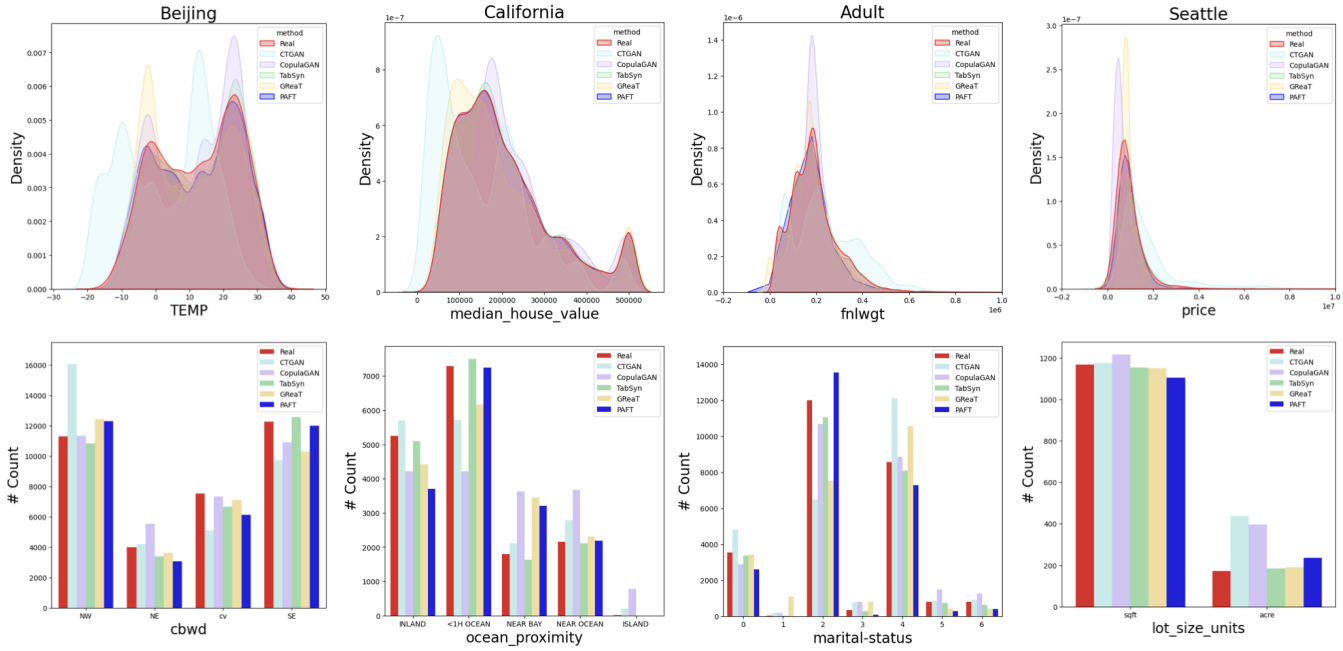


Figure 7: Column distributions visualization for each dataset generated by CTGAN, CopulaGAN, GReaT, and PAFT . The top row displays examples of numerical columns, while the bottom row presents examples of categorical columns. Overall, PAFT (Blue) has the closest distribution to real data (Red) compared to other synthesis methods. PAFT also showcase the ability to generate a wide range of diversity.

Admittedly, PAFT and all LLM fine-tuning based table generators share the identical challenge of time consuming (both training and sampling), comparing to classic DL or GAN based table generators. In exchange for the investment of time, there are several advantages to consider. These include the elimination of data preprocessing which is a significant time cost for a human expert in charge of data preparation, a deep understanding of real-world knowledge, and finally DL-bsaed generators have been shown to fail when the table generation task involves generating columns with textual sentences as their values.

D.6 Case Study: Influence of Statistical and Semantic Factors

Statistical Factor The occurrence of functional dependency in a table is influenced by various factors, including the number of rows and columns, the distinct values in the columns, the relationship between columns, and the presence of duplicate rows, etc. Table 1 and 5 shows different dataset may have different level of statistic difficulties.

Semantic Factors The acquisition of semantic factors is typically not achievable from direct observation of the data's appearance. Typically, this implies that the data's worth will be influenced

Data Privacy - Sniff Test: ML Discriminator Accuracy (Ideal → 50%)					
Method	**CTGAN**	**CopulaGAN**	**TabSyn**	**GReaT**	**PAFT**
Beijing	99.16± 0.08%	98.69± 0.39%	50.97±0.06%	51.1± 0.08%	**50.09± 0.05%**
US-locations	99.94± 0.03%	97.74± 0.22%	51.97±0.18%	50.47± 0.07%	**50.01± 0.01%**
California	98.35± 0.2%	86.64± 0.67%	50.64±0.15%	53.74± 0.27%	**49.89± 0.03%**
Adult	94.43± 0.53%	59.82± 0.9%	51.64±0.14%	**51.12± 0.26%**	48.75± 0.03%
Seattle	87.61± 1.06%	85.7± 2.0%	**50.12±0.9%**	68.27± 1.34%	47.21± 0.48%
Travel	77.96± 1.4%	74.14± 1.64%	**50.66±1.97%**	62.49± 1.2%	48.18± 0.81%

Table 13: Discriminator Performance (%): Comparison of synthesized data from CTGAN, CopulaGAN, TabSyn, GReaT, and PAFT . The scores stand for the accuracy for detecting real or fake data, where the ML models are trained using 50% real data and 50% random data. An ideal accuracy score is 50, indicating the model cannot distinguish between real and synthesized data. The best results are marked in bold, the second-best results are underlined.

	CTGAN		**CopulaGAN**		**TabSyn**	
	Training Time	Sampling Time	Training Time	Sampling Time	Training Time	Sampling Time
Adult	50:38 sec	0:47 sec	11:26 min	1:04 sec	46.35 min	7.38 sec
Beijing	55:72 sec	1:07 sec	13:94 min	2:67 sec	54:11 min	9:89 sec
California Housing	2:79 min	5:68 sec	5:72 min	1:02 sec	33:27 min	4:81 sec
US-locations	18:53 sec	0:17 sec	4:72 min	0:39 sec	28:6 min	4:47 sec
Seattle	3:56 sec	0:07 sec	24:93 sec	0:13 sec	18:43 min	0:64 sec
Travel	0:40 sec	0:04 sec	12:49 sec	0:06 sec	16:44 min	0:63 sec

	GReaT		**PAFT**	
	Training Time	Sampling Time	Training Time	Sampling Time
Adult	3:52 hr	37:47 min	3:49 hr	**5:15 min**
Beijing	4:10 hr	5:24 min	4:08 hr	5:21 min
California Housing	2:23 hr	4:16 min	2:22 hr	**2:58 min**
US-locations	55:48 min	58 sec	54:33 min	58 sec
Seattle	7:42 min	10 sec	7:43 min	11 sec
Travel	3:30 min	5 sec	3:33 min	5 sec

Table 14: A run time comparison of all generative models. Models were trained and fine-tuned using comparable hyper-parameters, and generated samples were of the same size as the real dataset. For certain datasets that pose difficulties for auto-regressive generation (such as Adult), PAFT can significantly enhance the quality of the generation process, resulting in reduced generation time. For typical datasets with fewer challenges, the time-efficiency performance of GreaT and PAFT is comparable. (The standard variance of time in the five random experiments was smaller than one secondary unit, so it has been omitted).

by real-world expertise in a specific field. For instance, map coordinates are influenced by the geopolitical borders of actual countries and states. Similarly, even if only a subset of data points from a mathematical function are observed, there is a need to comprehend the complete representation of that particular mathematical function.

Fig. 6 shows the difficulty of capturing the functional dependency can also leaded by the semantic conext of a sub-class in a mixture dataset, such as the state shape and geo location distribution. For this case, previous Table 2 have already shown the improvement of utilizing PAFT .

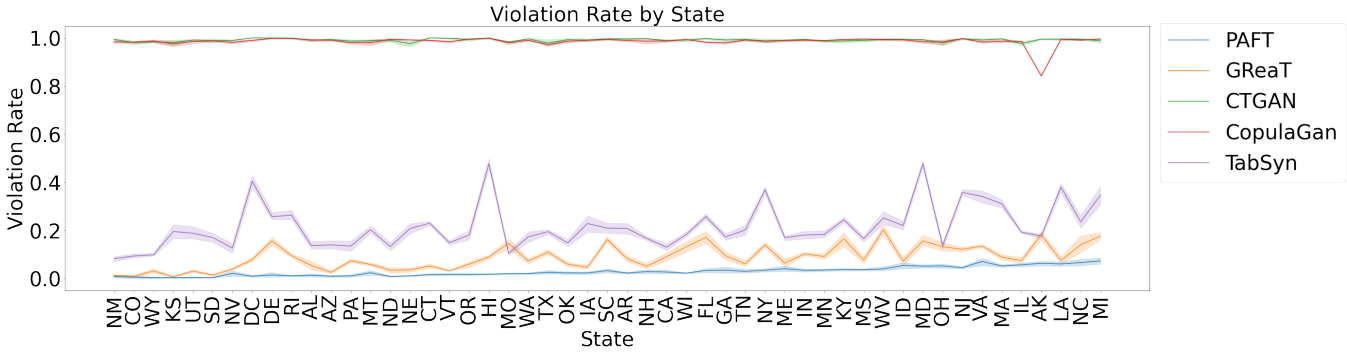


Figure 8: For a composite dataset, comparison of state-specific violation rates for different synthetic data generation approaches. Here, the states (x-axis) are sorted based on increasing violations. PAFT significantly mitigates state-specific violations in a composite dataset.