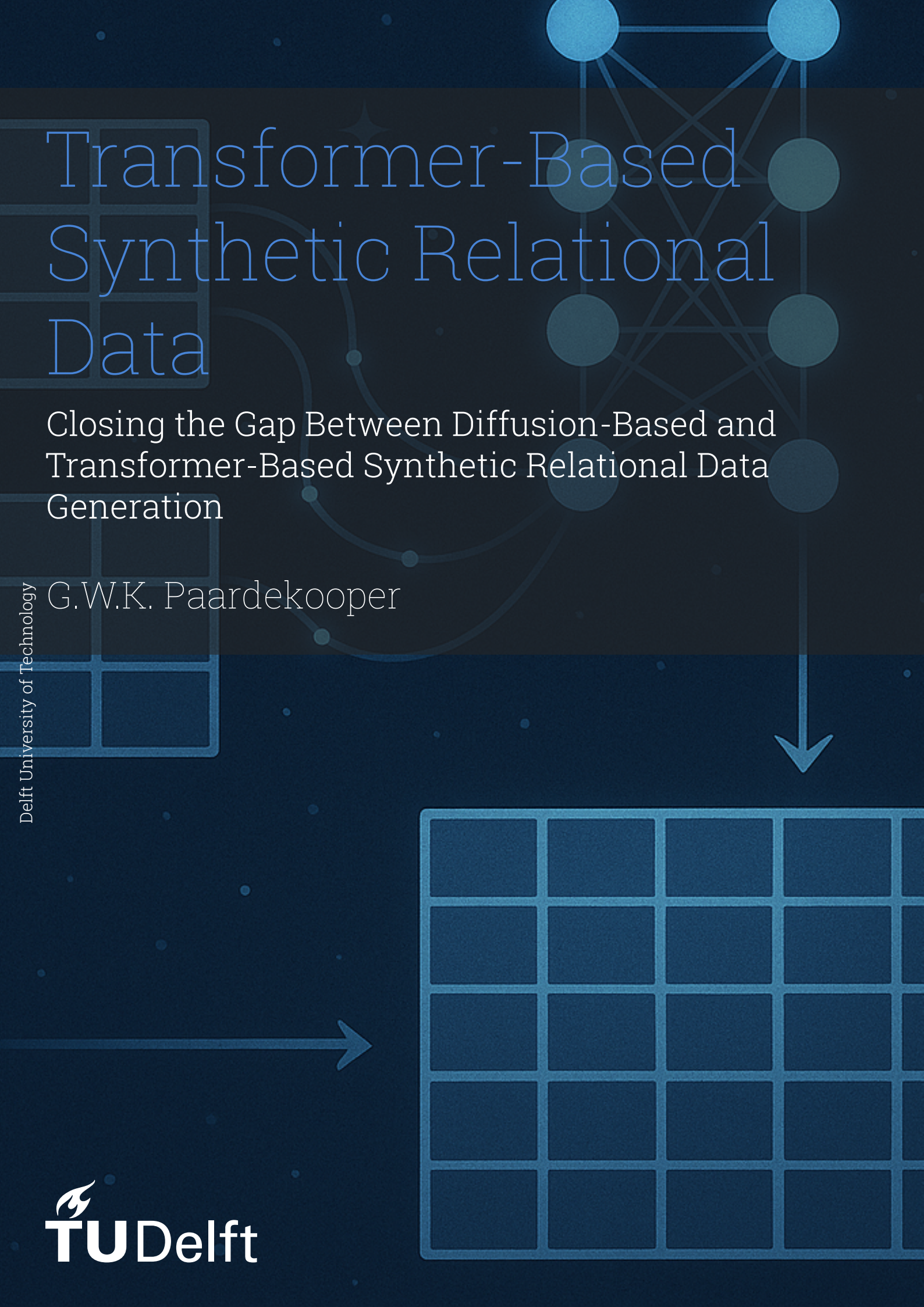
Transformer-Based Synthetic Relational Data

Closing the Gap Between Diffusion-Based and Transformer-Based Synthetic Relational Data Generation

G.W.K. Paardekooper

Delft University of Technology



TU Delft

Transformer-Based Synthetic Relational Data

Closing the Gap Between Diffusion-Based and Transformer-Based Synthetic Relational Data Generation

by

G.W.K. Paardekooper

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Friday July 11, 2025 at 10:00.

Student number:	4576918	
Project duration:	February 15, 2024 – July 11, 2025	
Thesis committee:	Dr. L. Y. Chen,	TU Delft, supervisor
	Dr. C. Lofi,	TU Delft
	MSc. J. M. Galjaard,	TU Delft

Cover:	generated by OpenAI's ChatGPT-o4-mini-high (May 29, 2025)
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

I would like to begin by sincerely thanking my thesis supervisor, Dr. Lydia Y. Chen, and my daily supervisor, Jeroen M. Galjaard, for their exceptional support and guidance throughout the completion of this thesis. There were times when I entered our weekly meetings with hesitation, especially when the experiments did not yield the results we wanted. Still, I always left feeling more motivated than when I arrived. Despite the many changes in direction, your encouragement remained consistent. The combination of your high-level, solution-oriented vision and deep technical expertise formed the backbone of this work, making our weekly meetings both enjoyable and productive. I'm also very grateful to Jeroen for his support, efforts, and detailed feedback while writing the research paper included in this thesis.

I would also like to thank my manager, Cees, for his support and flexibility in allowing me to combine a full-time job with this academic pursuit. His understanding and trust made it possible for me to stay committed to both.

To my friends: thank you for your understanding and encouragement throughout this journey. Even though we couldn't spend as much time together as we would have liked, your continued support and the good times we did have kept me going. I'm looking forward to celebrating with you and catching up properly now that this chapter is complete.

Most importantly, I would like to thank my family, loved ones, and those closest to me for their support over the years; I'm genuinely grateful to you all. To my mother, the most caring and hard-working person I know, and my sister, an endless source of laughter and joy, thank you for your advice, love, and unwavering support. I don't often say it out loud, but I'm so proud of you both. And to my girlfriend, Els, thank you for your patience, love, your words of encouragement, and your genuine interest in what I was working on, even when my explanations made little sense. You reminded me of what really matters and helped me push through.

In loving memory of my father, who taught me the value of hard work, to never give up, and inspired me to be curious, traits that have greatly helped me while working on this thesis. His incredible optimism, even in difficult times, showed me there's always a way forward. Above all, he was, and will continue to be, a fantastic dad whom I truly look up to.

G.W.K. Paardekooper
Delft, July 2025

Abstract

Data sharing for research and industrial applications faces significant challenges due to privacy constraints and regulatory requirements, driving the need for high-quality synthetic alternatives. Recent advances in synthetic data generation have demonstrated considerable success for single-table datasets, with emerging research extending these capabilities to multi-table *relational scenarios*. While transformer and diffusion architectures achieve state-of-the-art performance in single-table generation, a notable performance gap emerges when applied to relational data, where diffusion approaches consistently outperform transformer-based methods.

This thesis examines the factors contributing to this performance difference, conducting an evaluation using multiple baselines across both single and relational tabular datasets, with REaLTabformer and ClavaDDPM as state-of-the-art transformer- and diffusion-based approaches, respectively.

Our investigation reveals that the performance can mainly be attributed to the inadequate processing of contextual relationships and suboptimal strategies for representing inter-table dependencies in transformer-based models. To close this gap, we introduce two changes for transformer-based models: *layer sharing* to enhance parameter utilization and *contextual encoding* to better preserve the relational structure. These changes provide insight into the key design principles behind effective synthetic relational data generation using transformer-based models, particularly the need for architectures that account for context and facilitate practical knowledge transfer. The proposed methods result in substantial performance improvements, with gains of $1.52\times$ in Logistic Detection and $1.94\times$ in the Discriminator Measure metric.

Contents

1

Introduction

Data-driven research and industry increasingly depend on high-quality data to train machine learning models and perform statistical analyses. However, the sharing and utilization of real-world data face significant challenges, particularly in sensitive domains where privacy regulations, such as the GDPR [23], and data scarcity issues [6] limit access to real datasets. These constraints have increased the interest in synthetic data generation as a privacy-preserving alternative that maintains statistical properties while eliminating direct links to real individuals or entities. To this end, numerous deep-learning-based generative models have been developed to generate synthetic data.

These generative approaches have undergone substantial evolution in recent years, with the focus shifting from traditional statistical methods to deep-learning-based architectures. While significant progress has been achieved in generating synthetic data for individual tables [74, 44], the challenge becomes considerably more complex when dealing with multi-table relational datasets where preserving inter-table relationships and maintaining referential consistency are paramount. Recent research has demonstrated varying levels of success across different architectural approaches for synthesizing relational data.

Comparative studies suggest that diffusion-based architectures consistently outperform token-based transformer architectures in both single- and multi-table scenarios [39, 22], despite transformers offering advantages in terms of preprocessing requirements and being more easily integrated with other models or systems. In the remainder of this thesis, we use the terms ‘diffusion’ and ‘transformer’ as abbreviations for these respective architectures. The observed performance gap between diffusion-based and transformer-based models raises questions about the architectural choices and data representation strategies that impact the quality of synthetic data.

1.1. Problem Statement

This thesis addresses the consistently lower performance of transformer models compared to diffusion models. Our investigations suggest that this performance gap stems from fundamental architectural differences in how these models approach generating relational data. Understanding and addressing these limitations presents an opportunity to enhance transformer-based approaches for generating relational synthetic data.

1.2. Research Questions

Following the preceding problem statement, we aim to investigate and close the performance gap between diffusers and transformers. As such, to investigate the performance differences between relational data generation methods, we ask the following three questions:

- **RQ1**: How does the representation of relational data affect the performance of generative models? This question provides a systematic comparison of different data representation approaches, examining their impact on generation quality and computational requirements.
- **RQ2**: How can parameter sharing help narrow the performance gap between transformer-based and diffusion-based models for relational tabular data generation? This question explores whether sharing parameters across layers can enhance the effectiveness of our baseline transformer model.
- **RQ3**: How does the representation of relational cues affect the quality of transformer-based relational data? This question examines various approaches to incorporating contextual cues that enable transformer models to understand and preserve the hierarchical relationships inherent in multi-table datasets. Thereby investigating the impact of both their encoding and their position in the generative model's input.

1.3. Research Contributions

The primary contributions of this work include:

- A systematic analysis of performance disparities between transformer and diffusion architectures in both single-table and multi-table generation scenarios.
- Novel architectural enhancements for transformer-based models that substantially improve their relational data generation capabilities at a low computational cost.
- A comprehensive evaluation comparing different data representation strategies and their impact on data generation quality.
- Empirical demonstration of significant performance improvements, achieving up to 1.52–$1.94\times$ enhancement over baseline transformer approaches.

1.4. Report Structure

The remainder of this thesis is structured as follows: **Chapter 2** includes the written research paper part of this thesis, currently under review under the ACM's International Conference on Information and Knowledge Management (CIKM) 2025. In this work, we propose two methods for improving the performance of transformer-based synthetic relational data models. **Chapter 3** presents additional background for synthetic data generation, reviews prior art for both single-table and relational data. Additionally, it offers a technical background on transformer architectures. Moreover, this chapter also includes the specific application domains and detailed model comparisons that establish the context for our research. **Chapter 4** introduces the proposed techniques with extended analyses from the experiments in the research paper, along with various other experiments that provide insights into enhancements for generative models and identify which tricks did not yield any performance gains. **Chapter 5** summarizes the key findings of this thesis and outlines directions for future work in relational synthetic data generation.

2

Research Paper

In this chapter, we present the paper titled *"Minding the Gap: Improving Transformer-Based Relational Data Synthesis."* which is currently under review for the Applied Research Paper track at the 34th ACM International Conference on Information and Knowledge Management (CIKM 2025). The conference is scheduled to take place from November 10 to 14 in Seoul, South Korea. The paper investigates the gap in data fidelity and utility between diffusion- and transformer-based models for generating synthetic tabular data. We propose two simple methods that can be applied to existing transformer-based models, enabling them to outperform diffusion-based models on specific metrics.

Please note that due to a data processing error, some of the standard deviations of the logistic detection metric in Tables 2, 3, 5, and 6 are shown as 0.00. The corrected versions of these tables are provided in Appendix A.

Minding the Gap: Improving Transformer-Based Relational Data Synthesis.

Gijs Paardekooper
Cross Options
Amsterdam, The Netherlands
gijs.paardekooper@crossoptions.nl

Jeroen Galjaard
Delft University of Technology
Delft, The Netherlands
j.m.galjaard@tudelft.nl

Lydia Chen
Delft University of Technology
Delft, The Netherlands
lydiaychen@ieee.org

Abstract

Shareable tabular data is of high importance in industry and research. While generating synthetic records is well studied, research has only recently extended towards *relational data synthesis*. In the tabular generation setting, diffusion and transformer models boast superior performance over prior art. However, in the relational setting, diffusion models outperform transformers. This work focuses on the apparent performance gap between tabular transformers and diffusion models in the single (tabular) and multi-table (relational) settings, using REaLTabformer and ClavaDDPM as representative state-of-the-art models. We evaluate these architectures on a set of representative single and multi-table datasets, where we highlight the root causes for the gap between the methods. In our experiments, we attribute this difference to contextual information and data representation. To close this gap in the relational setting, we propose using two seemingly simple strategies: layer sharing and contextual cues. This work provides insights into the key design considerations for single- and multi-table generative models: incorporating contextual information and reusing existing knowledge. With the proposed methods, we achieve improvements of 1.52× and 1.94× for the Logistic Detection and Discriminator Measure metrics, respectively.

Keywords

Synthetic Tabular Data, Multi-Tabular, Transformer, Diffusion, Layer Sharing, Clustering

ACM Reference Format:
Gijs Paardekooper, Jeroen Galjaard, and Lydia Chen. 2025. Minding the Gap: Improving Transformer-Based Relational Data Synthesis.. In *34th ACM International Conference on Information and Knowledge Management (CIKM '25).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The increasing need for high-quality synthetic tabular data has spurred the development of many generative deep learning models over recent years. Following earlier seminal works from the image domain has led to Variational Auto-encoder (VAE) and GAN-based approaches: TVAE and CTGAN [31]. Similarly, advances in graph generation [22] and data-diffusion models [11] made grounds for methods like GOGGLE [17] and TabDDPM [15]. While these methods have provided means for ever-increasing synthetic data fidelity, they have primarily focused on flat datasets, where singular rows represent data entries. Extending this towards multiple *related* tables is a recent advance backed by diffusion and transformer architectures. ClavaDDPM [20] (Clava) builds on TabDDPM to synthesize (multi-)relational data. In a similar vein, developments from the natural language domain [27] have been successfully applied for tabular generation [4]. REaLTabFormer (RTF) [24] builds upon this idea to extend the data generation to the relational data domain.

While tabular generators have been well studied in the context of flat data [1, 23, 26], it naturally follows to consider their ability to create relational data. To do so, such models must not only learn to generate distinct tables, but also the relation between them—as depicted in Figure 1.Comparing the different models on both flat and (multi-)relational data provides insight into design considerations for applied research. Related work, SynthRela [13] introduces an evaluation framework for comparing and measuring the performance of relational tabular generative models. In this work, we build upon their findings that diffusion models consistently outperform transformer-based models for synthesizing relational data.

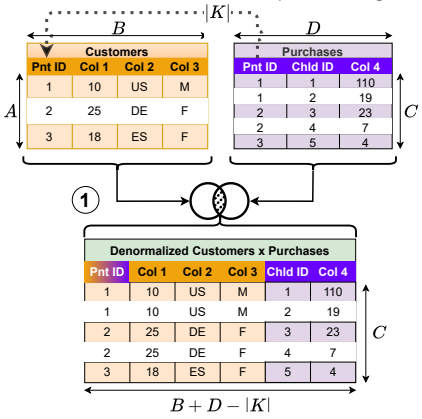


Figure 1: Example relational dataset with 'customers' as Parent (Pnt) table and 'purchases' as subsequent Child (Chld) table. ① illustrates creating a denormalized dataset by performing an inner-join on the primary key K 'Pnt ID', to flatten relational data. The pairs A, C and C, D, represent the table height and width, respectively. Note that the denormalized dataset contains duplication as each Child table's record is prefixed with its corresponding Parent row.

CIKM '25, Seoul, South Korea

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Tabular evaluation on Machine Learning Efficiency (MLE), Discriminator Measure (DM) scores, and train time. Scores reported as mean$_{\pm\text{standard deviation}}$ with best results in boldface and second best underlined highlighted. For both metrics, lower is better. We denote training time in seconds with T_{train}.

		Orig. Data	GAN	CTGAN	TVAE	GOGGLE	GReaT	Clava	RTF	OURS
Breast Cancer [29]	MLE	$0.05_{\pm 0.02}$	$0.07_{\pm 0.03}$	$0.05_{\pm 0.01}$	$\mathbf{0.04_{\pm 0.01}}$	$0.08_{\pm 0.03}$	$0.08_{\pm 0.03}$	$\mathbf{0.04_{\pm 0.01}}$	$\mathbf{0.04_{\pm 0.01}}$	$\mathbf{0.04_{\pm 0.02}}$
	DM	$0.03_{\pm 0.02}$	$0.72_{\pm 0.10}$	$0.30_{\pm 0.08}$	$0.25_{\pm 0.10}$	$0.45_{\pm 0.05}$	$0.43_{\pm 0.12}$	$\mathbf{0.09_{\pm 0.04}}$	$0.22_{\pm 0.06}$	$0.19_{\pm 0.07}$
	T_{train}	-	1448	1209	905	2739	322	181	128	**83**
Cali. Housing [19]	MLE	$0.35_{\pm 0.09}$	$0.41_{\pm 0.06}$	$0.49_{\pm 0.07}$	$0.45_{\pm 0.08}$	$0.75_{\pm 0.05}$	$0.42_{\pm 0.09}$	$\mathbf{0.32_{\pm 0.08}}$	$0.36_{\pm 0.08}$	$0.37_{\pm 0.08}$
	DM	$0.01_{\pm 0.00}$	$0.46_{\pm 0.30}$	$0.50_{\pm 0.27}$	$0.56_{\pm 0.27}$	$0.78_{\pm 0.09}$	$0.22_{\pm 0.05}$	$\mathbf{0.06_{\pm 0.03}}$	$0.16_{\pm 0.04}$	$0.18_{\pm 0.04}$
	T_{train}	-	732	1729	848	425	8407	**379**	652	659
Adult [2]	MLE	$0.10_{\pm 0.00}$	$0.16_{\pm 0.03}$	$0.13_{\pm 0.01}$	$0.13_{\pm 0.01}$	$0.16_{\pm 0.01}$	$0.11_{\pm 0.01}$	$\mathbf{0.10_{\pm 0.00}}$	$\mathbf{0.10_{\pm 0.00}}$	$\mathbf{0.10_{\pm 0.00}}$
	DM	$0.00_{\pm 0.00}$	$0.99_{\pm 0.01}$	$0.89_{\pm 0.08}$	$0.77_{\pm 0.04}$	$0.99_{\pm 0.01}$	$0.28_{\pm 0.04}$	$\mathbf{0.06_{\pm 0.03}}$	$0.12_{\pm 0.03}$	$0.15_{\pm 0.04}$
	T_{train}	-	2777	431	**335**	1082	10704	401	1087	1082
Magic [3]	MLE	$0.11_{\pm 0.01}$	$0.13_{\pm 0.02}$	$0.16_{\pm 0.02}$	$0.15_{\pm 0.02}$	$0.14_{\pm 0.02}$	$0.13_{\pm 0.02}$	$\mathbf{0.11_{\pm 0.02}}$	$\mathbf{0.11_{\pm 0.01}}$	$0.12_{\pm 0.02}$
	DM	$0.01_{\pm 0.01}$	$0.94_{\pm 0.09}$	$0.82_{\pm 0.23}$	$0.86_{\pm 0.17}$	$0.47_{\pm 0.20}$	$0.34_{\pm 0.05}$	$\mathbf{0.04_{\pm 0.03}}$	$0.11_{\pm 0.03}$	$0.13_{\pm 0.05}$
	T_{train}	-	1441	191	**143**	6601	5113	360	1688	1643
Shoppers [21]	MLE	$0.09_{\pm 0.01}$	$0.10_{\pm 0.01}$	$0.09_{\pm 0.01}$	$0.09_{\pm 0.01}$	$0.12_{\pm 0.00}$	$0.11_{\pm 0.02}$	$\mathbf{0.08_{\pm 0.01}}$	$0.09_{\pm 0.01}$	$0.09_{\pm 0.01}$
	DM	$0.01_{\pm 0.00}$	$0.97_{\pm 0.06}$	$0.97_{\pm 0.04}$	$0.93_{\pm 0.09}$	$0.95_{\pm 0.06}$	$0.33_{\pm 0.06}$	$0.59_{\pm 0.01}$	$\mathbf{0.20_{\pm 0.06}}$	$0.27_{\pm 0.08}$
	T_{train}	-	740	367	592	622	3684	**335**	372	427

Moreover, the diffusion models enjoy favourable training and inference time-cost over their transformer counterparts. However, the transformer-based approaches are favourable due to their limited requirements on data-preprocessing [4, 24].

These aforementioned results lead us to quantitatively analyse Clava and RTF in single-table and relational tabular settings. Our preliminary results spurred the following question: How does the representation of relational records influence the fidelity of data generated by generative models? To further illustrate this point Figure 1 shows how we can convert a parent table (with A rows and B columns) and a child table (with C rows and D columns) into one flat table—i.e., denormalized. The parent and child tables are linked through a key—'Pnt. ID'—with K denoting the features making up this key. We aim to investigate whether we can learn these relations when represented as a flat table.

Based on the performance gap analysis, we unveil that the existing transformer model lags behind the diffusion model because of the suboptimal modeling on the contextual information and parent-child tables We thus propose novel strategies, such as layer sharing and contextual information, to close the performance gap and improve the existing transformer-based tabular generative modes. Our design and evaluation center on two sub-research questions: (i) how can layer sharing reduce the performance gap, and (ii) how does contextual information impact transformer-based relational models? In our evaluations, we improve up to 1.5–1.9× in terms of Logistic Detection (LD) and Discriminator Measure (DM), respectively, over baseline REaLTabFormer performance. Thus demonstrating our design to be effective for closing or bridging the gap between transformers and diffusion models in relational tabular data modeling.

2 Related Work and Background

Our observations in the single and relational settings ignite the question of what causes this performance difference. Whether the difference is caused by the methods, architectures, or how inter-row relations are modeled. As such, we first consider the related works

Single Table Generation synthesis has been well studied, following trends from the image and language generation domains. Initial models in the tabular data setting consisted of GAN and VAE derivatives, adapted to work on tabular data [31]. CTGAN builds upon the regular GAN architecture to work with heterogeneous data and handle class imbalance. More recent advances include the usage of graph-based models like GOGGLE [17], diffusion as TabDDPM [15] and Clava [20], and natural-languag derived models as GREaT [4] and REaLTabFormer [24]. While GReAT and REaLTabFormer share the same underlying backbone, i.e. Transformer [27], they differ in their modeling approach. The former advocates using the 'semantic information' of tables, by fine-tuning a pre-trained language model and converting samples into natural text. REaLTabFormer, provides superior performance by using only the architecture and a novel tokenization.

Relational Table Generation extends the prior setting, seeing recent advancements with RTF and Clava. Although both generate relational data, they differ significantly in key aspects. RTF is an auto-regressive-based model with a relational sequence-to-sequence (Seq2Seq) head, which we call 'relational adapter'. Herein, the relational adapter uses the learned parent table model as the encoder. Thus, sequentially learning the parent and child, using the former to guide the relational sampling. Clava, on the other hand, achieves relational sampling through condition-guided diffusion sampling. Thus making the learning independent of the per-table models. Instead of encoding the relations explicitly, Clava learns to condition based on a learned data-clustering approach. Therein

assigning 'cluster IDS' to records, on which a classifier guidance model is used to generate samples a low-dimensional bottleneck. As a results considerably simplifying the conditional generation of Child records. Instead requiring child samples to be sampled from a class-dependent distribution, as proxy for the parent. Effectively creating an information bottleneck during the training and sampling process of the Child rows. As SyntRela [13] shows that Clava outperforms RTF on their benchmark, this motivates further investigation into the performance gap between the two models. This is particularly relevant given that Transformer-derived architectures have demonstrated strong performance on relational and structured sequence-to-sequence tasks [16].

3 Evaluating the Performance Gap

Before moving towards our method's considerations, we create a match-up between related work for single (Table 1) and relational data generation (Table 3). In the single table case, Clava consistently achieves best or second-best results in Discriminator Measure (DM) and Machine Learning Efficiency (MLE), respectively—measures for data fidelity and utility, as discussed below. RTF achieves similar performance, ranging from a close second-place to trading blows with Clava. Contrary to our expectations, the minimal gap between RTF and Clava widens considerably when we move to the relational tables, as shown in Table 2 'Baseline' column. In this setting, Clava's Logistic Detection (LD) and Discriminator Measure (DM) often far exceed those of REaLTabFormer—metrics representing the difficulty of distinguishing between real and generated data. Following this salient result, we investigate the apparent performance gap between RTF and Clava.

Data Utility. To evaluate data utility, we use downstream tasks to assess the quality of synthesized datasets. To this end, we employ Machine Learning Efficacy (MLE) [32] on single table datasets, which represents the performance of a classical machine learning (ML) model trained on synthetic samples and evaluated on real test samples. As the choice of downstream model can impact reported quality, we use five ML models per dataset: Decision Tree (DT) [6], Random Forest (RF) [5], Gradient Boosting [9], AdaBoost [8], and XGBoosted Random Forest [7]. We report an aggregate score of the models mentioned above for every dataset over multiple runs and replications. Scores are represented as the arithmetic mean and standard deviation.

Data Fidelity. Similar to data utility, more realistic data should match the original distribution. We evaluate data fidelity using the Discriminator Measure and extend this with the Logistic Detection [24] in the relational setting. Both tasks evaluate the discernibility of real and fake data, differing in their underlying task and model selection. The Logistic Detection trains a Random Forest Classifier—following the approach of RTF—to discriminate between real and generated records, thus providing a means to pick up on broad-scale inconsistencies between real and fake data. These two metrics jointly provide insight in how well the generated relational match the original Parent and Child table. We report the a normalized Discriminator Metric (DM), computed as $\mathrm{DM}' = |\mathrm{DM} - 50|/50$. For notational convenience, refer to the normalized Discriminator Metric when we refer to DM'. DM scores are computed based on the model's predictive accuracy, whereas LD scores are computed based on the ROC-AUC measure, with scores in the range of 0 to 100 in line with RTF.

Table 2: Relational-dataset evaluation on Discriminator Measure (DM), Logistic Detection (LD) scores, and train duration. Scores are reported as mean$_{\pm\text{standard deviation}}$. For DM, lower is better; for LD, higher is better. Best scores are highlighted in boldface and second-best are underlined. We denote training time in seconds with T_{train}. Out of Memory (OOM) indicates an inability to complete with the allotted resources.

		Metric	Original	Baseline		Flattened	
				RTF	Clava	RTF	Clava
AirBnB	Parent	LD	98.98$_{\pm 1.00}$	52.63$_{\pm 0.00}$	**85.92$_{\pm 0.00}$**	29.24$_{\pm 0.00}$	71.45$_{\pm 0.00}$
		DM	0.02$_{\pm 0.01}$	**0.57$_{\pm 0.05}$**	**0.57$_{\pm 0.05}$**	0.98$_{\pm 0.03}$	1.00$_{\pm 0.00}$
	Child	LD	91.37$_{\pm 1.21}$	42.22$_{\pm 0.00}$	**89.05$_{\pm 0.00}$**	62.30$_{\pm 0.00}$	76.52$_{\pm 0.00}$
		DM	0.05$_{\pm 0.02}$	0.44$_{\pm 0.05}$	0.54$_{\pm 0.07}$	**0.28$_{\pm 0.04}$**	0.88$_{\pm 0.03}$
		T_{train}	-	2762	1049	8891	**407**
Rossman	Parent	LD	95.69$_{\pm 0.00}$	83.58$_{\pm 0.00}$	21.78$_{\pm 0.00}$	**94.42$_{\pm 0.00}$**	82.59$_{\pm 0.00}$
		DM	0.85$_{\pm 0.01}$	0.54$_{\pm 0.02}$	**0.09$_{\pm 0.04}$**	0.49$_{\pm 0.01}$	0.72$_{\pm 0.02}$
	Child	LD	95.19$_{\pm 0.00}$	57.42$_{\pm 0.00}$	88.01$_{\pm 0.00}$	**90.84$_{\pm 0.00}$**	75.51$_{\pm 0.00}$
		DM	0.84$_{\pm 0.05}$	0.40$_{\pm 0.05}$	0.10$_{\pm 0.04}$	**0.09$_{\pm 0.00}$**	0.62$_{\pm 0.06}$
		T_{train}	-	1754	1119	3035	**384**
FTP	Parent	LD	99.12$_{\pm 0.00}$	85.16$_{\pm 0.00}$	**97.54$_{\pm 0.00}$**	OOM	95.74$_{\pm 0.00}$
		DM	0.02$_{\pm 0.01}$	0.11$_{\pm 0.02}$	**0.06$_{\pm 0.04}$**	OOM	0.09$_{\pm 0.00}$
	Child	LD	92.31$_{\pm 0.00}$	29.20$_{\pm 0.00}$	86.55$_{\pm 0.00}$	OOM	**88.02$_{\pm 0.00}$**
		DM	0.49$_{\pm 0.04}$	0.73$_{\pm 0.02}$	0.56$_{\pm 0.02}$	OOM	0.48$_{\pm 0.00}$
		T_{train}	-	1561	961	-	**366**

As such, evaluate the methods in a 'flattened' relational setting—by denormalizing tables through inner-joining based on their primary key. We train both methods on the flattened table, of which we report their performance in the 'Flattened' column of Table 2. The resulting flat table contains redundant information where parent table attributes are replicated across multiple rows corresponding to each associated child record, effectively flattening the hierarchical relationship into a singular tabular format. While this denormalisation process increases storage requirements and introduces data redundancy, it potentially simplifies the data structure enabling the application of single-table synthesis methods. Herein, we note that the FTP results are excluded for REaLTabFormer as the method failed to compute within the imposed 24 GB memory constraint. While Clava's Child performance is mostly negatively affected, REaLTabFormer shows a remarkable improvement in data quality. Matching or exceeding Clava's baseline performance on the Child table regarding DM for AirBnb and Rossmann.

Meanwhile, we also observe that overall Parent performance is negatively impacted. Herein, there is considerable degradation on Airbnb and only marginal improvements on Rossman. In addition, flattening the data brings considerable overhead for RTF,

now requires storing all activations of the values of both the parent and child rows. Contrary to the Seq2Seq approach, which reduces this overhead by training them sequentially. This performance and overhead impact support the explicit relational data modeling in a model's architecture. However, it gives a glimpse into closing the performance gap between REaLTabFormer and Clava through the implicit layersharing that 'Flattened' data imposes. The same model weights are used to generate the parent and child rows.

Although this heads us towards closing the gap, our results hint towards a simplified conditional relation that benefits the relational modeling. As Clava only Specifically, we highlight the differences in data representation and, more so, the provided *contextual information*. The former differs as Clava models this as a bottlenecked 'hop' versus the effectively 'flat' representation of RTF. Secondly, we say the 'contextual' information is provided to the model to sample relational data. By RealTabFormer's Seq2seq architecture, the relational sampler can see all of the parent's information. Clava, however, uses clustering and a classifier guidance method (on cluster ID) to steer relational sampling.

> **Summary:** Following our preliminary results, we show that Clava and RTF *exhibit a considerable performance gap* for generating relational data. Moreover, we investigate the impact of *representing the relational information*, by denormalizing the relational dataset as a single (flattened) table. Herein, we show that while both models benefit from modeling two tables as a relation over flattened, *RTF relational data performance improves considerably for its Child table.* Following these observations and results, we hypothesize that auto-regressive models can benefit from sharing their relational representation.

4 Method

Following our preliminary results and gained insights, we focus on leveraging these to close the performance gap. We propose to change the relational data representation to the base adapter and provide hints to the base and adapter models to simplify the conditional distribution of relational records. Towards the latter, we propose providing additional 'in-context' hints to the transformer model during the relational sampling. Towards the former, we consider altering the representation of relational information by creating a denormalized single table of the parent and child tables as illustrated in Figure 1. In Figure 2, we show the four steps our method proposes.

In short. First, in step ① we cluster on the joint parent (X) and child (Y) tables as shown in Algorithm 1. Then, we proceed in step ② by appending this information as a categorical column to X and Y, at column index q. We then tokenize the augmented training datasets and train our Base model's transformer on the tokenized parent table data X. Finally, in step ④ we apply layer sharing between the Base model and the Decoder model of the Relational Adapter, while fixing their parameters. As both models now inherit their parameters from the Base model, only the Adapter model's Child vocabulary can be optimized on the Child's data.

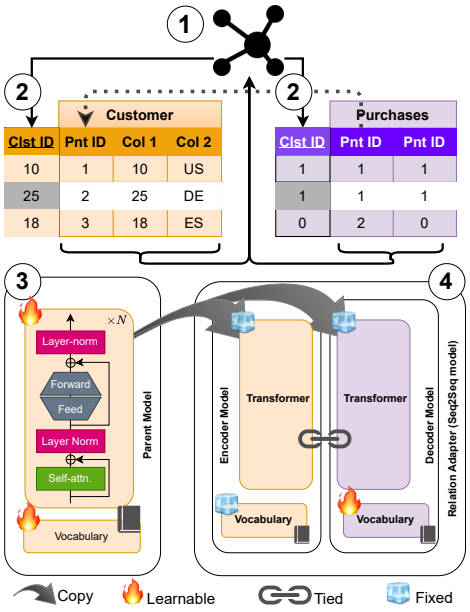


Figure 2: Representing tabular data with contextual information, orange and purple indicate Parent and Child data/model respectively. Contextual information is added to related Parent and Child records using clustering ①–②. Subsequently a Base model is trained on the Parent Table (③), which is subsequently used ④ as a frozen Encoder-Decoder for learning the relational Adapter. Therein sharing the Parent table's model (Encoder) layers with the relational (Decoder) of the relational adapter sequence-to-sequence (Seq2Seq) model, while only training the *encoder's vocabulary*.

In-Context Relational Cues. As shown in Figure 2 steps ①–②, we add contextual information to the training data. Provided with Clava's relational performance, we consider the impact of its relational modelling. Given that classifier-based diffusion on its own does not translate to an auto-regressive architecture, we turn to in-context representation. Similar to classifier based guidance, this makes the conditional 'guidance' token learnable before being used during inference. Therewith providing a signal to the generator to conditionally sample from a simplified distribution. Hereby, both letting the generator learn to steer the generation process conditioned on the cluster ID token. Subsequently, allowing to conditionally sample from the cluster ID by providing the model with the cluster ID as context As a result, providing *classifier-based guidance* to the auto-regressive generative model.

To create the contextual information to the model, we augment the training data with a Cluster ID (Clst ID in Figure 2). We add these cluster IDs to both the parent and child table with the location of the cluster ID being configurable, akin to Clava's Classifier Guidance approach. Thereby providing a piece of categorical information, matching the model's tokenizer discretization process. For a fair comparison with Clava, we share the obtained clustering

Algorithm 1 Cluster-Augmented Model Training. For the reader's convenience, we refer with ①–④ to the steps in Figure 2.

Require: Training dataset D_{train}, clustering algorithm `Cluster`, number of clusters k, randomly initialized Base model, randomly initialize Adapter model, Tokenization function `Tokenize`, context insertion index q.

Encode dataset
$\mathbf{X}, \mathbf{Y} \leftarrow D_{\text{train}}$ {Split Parent and Child.}
Step ①: Cluster encoded data
$\mathbf{C}_X, \mathbf{C}_Y \leftarrow \texttt{Cluster}(\mathbf{X}, \mathbf{Y}, k)$ {Get cluster assignments for Parent and Child.}
Step ②: Add contextual information
{Insert context for rows $\mathbf{X}, \mathbf{Y}$ with cluster ID from step ① at position q}
for $o = 1$ **to** $|\mathbf{X}|$ **do**
  $\mathbf{X}[i] \leftarrow \text{insert}(\mathbf{C}_X[i], \mathbf{X}[i], q)$
end for
for $j = 1$ **to** $|\mathbf{Y}|$ **do**
  $\mathbf{Y}[j] \leftarrow \text{insert}(\mathbf{C}_Y[j], \mathbf{Y}[j], q)$
end for
Step ③: Train base-model
$\mathbf{X} \leftarrow \texttt{Tokenize}(X)$
Base $\leftarrow$ Train(Base, $\mathbf{X}$)
Step ④: Train Relational 'Adapter'
Adapter $\leftarrow$ Base {Link Base & Adapter Transformer block weights}
$\mathbf{X} \leftarrow \text{Base}(\mathbf{X})$
$\mathbf{Y} \leftarrow \texttt{Tokenize}(\mathbf{Y})$
Adapter $\leftarrow$ Finetune(Adapter, $\mathbf{X}, \mathbf{Y}$) {Only fine-tune Adapter model's vocabulary.}
Returns: Base, Adapter

IDs between methods. In our experiments we run the clustering algorithm on data with one-hot encoding for categorical features, to enable clustering in data-space.

Layer sharing. After preparing the Parent and Child tables with contextual information, we continue by training the generative model. Before continuing, we recall that RTF learns a decoder-only 'sequence-to-sequence' model. By first training an encoder on the 'parent' output, it learns to represent a representation for a secondary model. This latter model, is then learned separately to predict a child 'completion' given a parent's records encoded representation. Of which the Parent tables Base model acts as an encoder for the Parent module, to create a latent representation of a parent's table records. Initially, these models both start off randomly initialized. Subsequently, we tokenize the Parent and Child dataset on which the models are trained sequentially. First we train the Base model on the augmented Parent, in step ③. Afterwards, we continue by fine-tuning the Adapter module.

To then learn the relational Adapter we reduce the number of learnable parameters compared to REaLTabFormer, as show ④ in Figure 2. We do so by performing layer-wise parameter tying between the Base and Adapter model . More formally, let each Base and Adapter learnable layer $L_{(\cdot)}$ be indexed by $p, c \in [0, N-1]$, then $\forall\ p, c\ s.t.\ p \neq c \implies L_p = L_c$. Moreover, we freeze the Parent and Child module, optimizing only the tokens to represent the Child rows. As a result, we are left with around $\sim$ 1% of the learnable parameters during the fine-tuning of the Adapter—compared to REaLTabFormer.

As such, leaving only the Child table's specific open to optimization during the fine-tuning of the Relational Adapter. As a result treating the relational data generation akin to a sequence-to-sequence task (Seq2Seq), such as sentence translation tasks. Herein, we remark that layer sharing in the NLP domain has been applied successfully [34, 35, 33, 30].

5 Evaluation

In this section, we continue with the evaluation of our proposed method. We evaluate the models on the same dataset as shown in section 3, on which we provide additional information in subsection 5.1. We consider the following three experimental setups to evaluate the proposed method, and ablate the design considerations. First, we evaluate the method on the single and relational datasets, comparing against REaLTabFormer and ClavaDDPM. Second, we ablate our design considerations by separating the evaluation of layer-sharing and contextual IDs. Last, we consider the impact of clustering approaches in providing more informed contextual information to the model. Before proceeding to the experimental results, let us detail the common experimental setup.

5.1 Experimental Setup

As shown prior in section 3 we evaluate baseline models on five single-table datasets; Breast Cancer Wisconsin (Breast) [29], California Housing (Housing) [19], Adult Income (Adult) [2], MAGIC Gamma Telescope (Magic) [3] and the Online Shoppers Purchasing Intention Dataset (Shoppers) [21]. We highlight key characteristics of these datasets in Table 4. Moreover, we consider the relational model's performance on three relational datasets: AirBnB New User Bookings [10], Rossmann Store Sales [14], and FTP [18]. These relational datasets consist of a parent and a child table linked through one primary key, as illustrated by the dotted line in Figure 1.

5.2 Narrowing the Relational Gap

First, we evaluate our proposed method in the relational and single table settings. We provide the single table results in Table 1 under the 'OURS' column. We can first conclude that providing a contextual cue to the model has a minor impact on the train duration over RTF, seeing similar training times, except for Shoppers, seeing an increase of $\sim$ 14%. Moreover, we also observe that context only minimally impacts the downstream utility regarding MLE compared to RTF. In terms of detectability, again, our method achieves a similar performance to that of RTF.

However, in the relational setting, as tabulated in Table 3, we observe a clear pattern in the delta between Clava and RTF on child table performance. Seeing that the gap is reduced or even switches in favour of our proposed method, coming from RTF compared to Clava. Seeing an improvement over all child-related metrics, narrowing the LD performance gap from $\sim$ 12–57 (and 0.3–0.23 for DM), to $\sim$ 12–9 (and for DM to −0.35–0.17). For the parent data, we see improvements over RTF, outperforming both baselines on AirBnB, and providing minor enhancements on Rossman and FTP.

Table 3: Multi-table relational dataset Logistic Detection (LD) and Discriminator Measure (DM) scores. Relational (Rel.) and Flat correspond to the original relational dataset and the flattened or denormalized (1NF) dataset, respectively. Metric shown as mean$_{\pm\text{standard deviation}}$. For DM, lower is better; for LD, higher is better. Bold face indicates best result and underlined second best. We denote training time in seconds with T_{train}. DNC denotes 'did not compute' as they resulted in OOM regardless of batch-size. We denote the time required to train the models by T_{train}.

		Metric	Original	RealTabFormer	Clava	OURS	Ablation* (subsection 5.3)	
							Layer Share	Context ID
AirBnB [10]	Parent *(Users)*	LD	98.98±1.00	52.63±0.00	85.92±0.00	**87.24±0.00**	51.55±0.00	**87.41±0.00**
		DM	0.02±0.01	0.57±0.05	0.57±0.05	0.17±0.06	0.57±0.04	**0.13±0.02**
	Child *(Sessions)*	LD	91.37±1.21	42.22±0.00	**89.05±0.00**	67.18±0.00	71.85±0.00	32.31±0.00
		DM	0.05±0.02	0.44±0.05	0.54±0.07	**0.33±0.07**	0.35±0.07	0.63±0.05
		T_{train}	-	2762	1049	3228	4347	4168
Rossmann [14]	Parent *(Stores)*	LD	95.69±0.00	83.58±0.00	21.78±0.00	**86.53±0.00**	85.65±0.00	**86.53±0.00**
		DM	0.85±0.01	0.54±0.02	**0.09±0.04**	0.53±0.01	0.54±0.02	0.53±0.01
	Child *(Sales)*	LD	95.19±0.00	57.42±0.00	**88.01±0.00**	79.30±0.00	64.97±0.00	77.24±0.00
		DM	0.84±0.05	0.40±0.05	**0.10±0.04**	0.27±0.07	0.37±0.06	0.27±0.07
		T_{train}	-	1754	1119	3514	1758	3517
FTP [18]	Parent *(Sessions)*	LD	99.12±0.00	85.16±0.00	**97.54±0.00**	85.54±0.00	88.44±0.00	81.35±0.00
		DM	0.02±0.01	0.11±0.02	**0.06±0.04**	0.11±0.02	0.10±0.02	0.13±0.02
	Child *(Products)*	LD	92.31±0.00	29.20±0.00	**86.55±0.00**	71.25±0.00	69.19±0.00	26.79±0.00
		DM	0.49±0.04	0.73±0.02	0.56±0.02	**0.21±0.02**	0.62±0.05	0.77±0.01
		T_{train}	-	1561	961	1454	1310	1596

5.3 Ablation: Evaluating Layer Sharing and Contextual Cues

Next, we consider the impact of the two design considerations brought forward in our method. Herein, we are ablating on the contribution towards improving relational transformer-based generation by evaluating the contribution of layer sharing (Layer Share) and the contextual cue (Context ID) independently on the relational datasets. We summarize the result in the 'Ablation' column of Table 3.

We observe similar trends to the 'implicit' layer sharing found in the data-flattening experiments from section 3. Augmenting the data with a context ID boosts RTF—except on FTP's 'products' table—and the *joint application* of contextual information and layer sharing consistently aids the generation of transformer-based relational data generation.

Table 4: Overview of datasets used, with names of the Parent and *Child* Tables for relational datasets.

	Dataset	Table	#Rows	#Columns		Task
				Cat.	Num.	
Single-table	Breast		569	1	30	Bin. Class.
	Housing		20,640	0	9	Reg.
	Adult		45,222	9	6	Bin. Class.
	Magic		19,020	1	9	Bin. Class.
	Shoppers		12,330	7	9	Bin. Class.
Relational	AirBnB	Users	10,000	13	3	Multi. Class.
		Sessions	191,025	5	1	
	Rossmann	Stores	1,115	3	7	Reg.
		Sales	68,015	3	6	
	FTP	Sessions	15,000	1	3	Bin. Class.
		Products	33,455	5	1	

5.4 Ablation: Contextual Cue Position

Here we evaluate the impact of placing the relational context cue at different positions to the model. Considering the first, second, or last column during step ② of our method (see section 4) before training. We recall that in the transformer architectures, tokens in early positions are attended to by all subsequent ones. As such, providing early context acts as learned 'contextual guidance', whereas providing it in a later position acts as an auxiliary training task—i.e., predicting the 'cluster' ID before generating subsequent child records. In Table 5 we report the results of altering the insertion index of the contextual cue to the transformer model. These results show that placing a token 'early', i.e., before the Parent ID, negatively impacts generation quality. We speculate that modeling the parent as conditioned on the contextual ID makes the distribution harder to generalize. As the training samples may lie near the decision boundaries of the clustering model, following step ①, requiring the model to 'commit' too early during generation to the conditional signal. In the results, column index 1 consistently

Table 5: Relational-dataset ablation on the impact of contextual information placement on Discriminator Measure (DM), Logistic Detection (LD) scores, and train duration. Idx. refers to the column index position at which the cluster IDs are inserted during ② of our method, where '1' is the default. Scores are reported as mean$_{\pm\text{standard deviation}}$. For DM, lower is better; for LD, higher is better. Best scores shown in boldface, second-best are underlined. We denote training time in seconds with T_{train}.

		Metric	Original	Insertion Indx.			
				RTF	Idx. 0	Idx. 1	Last Idx.
AirBnB	Parent	LD	98.98$_{\pm 1.00}$	52.63$_{\pm 0.00}$	29.24$_{\pm 0.00}$	**78.99$_{\pm 0.00}$**	47.56$_{\pm 0.00}$
		DM	0.02$_{\pm 0.01}$	0.57$_{\pm 0.05}$	0.98$_{\pm 0.03}$	**0.22$_{\pm 0.00}$**	0.62$_{\pm 0.00}$
	Child	LD	91.37$_{\pm 1.21}$	42.22$_{\pm 0.00}$	**62.30$_{\pm 0.00}$**	50.64$_{\pm 0.00}$	59.06$_{\pm 0.00}$
		DM	0.05$_{\pm 0.02}$	0.44$_{\pm 0.05}$	**0.28$_{\pm 0.04}$**	0.42$_{\pm 0.00}$	0.39$_{\pm 0.00}$
		T_{train}	-	2762	2823	**2004**	2575
FTP	Parent	LD	99.12$_{\pm 0.00}$	85.16$_{\pm 0.00}$	84.63$_{\pm 0.00}$	85.54$_{\pm 0.00}$	**85.60$_{\pm 0.00}$**
		DM	0.02$_{\pm 0.01}$	**0.11$_{\pm 0.02}$**	0.13$_{\pm 0.03}$	**0.11$_{\pm 0.02}$**	**0.11$_{\pm 0.01}$**
	Child	LD	92.31$_{\pm 0.00}$	29.20$_{\pm 0.00}$	57.96$_{\pm 0.00}$	**71.25$_{\pm 0.00}$**	**68.02$_{\pm 0.00}$**
		DM	0.49$_{\pm 0.04}$	0.73$_{\pm 0.02}$	0.60$_{\pm 0.03}$	**0.21$_{\pm 0.02}$**	0.26$_{\pm 0.00}$
		T_{train}	-	1561	1707	**1454**	1584

provides the best or second-best scores for the parent tables. The main improvement brought forward by contextual cues in general can be seen in the parent table. We thus focus mainly on the scores of the parent tables when evaluating the different indices used to place the context cues. For FTP, the scores are close with minor differences in the parent table performance. On the AirBnB dataset, we see different results, with the second position (index 1) showing the best performance for the parent table.

5.5 Ablation: Clustering Representation

Lastly, we consider the impact of altering the data representation on which the clustering algorithm is run. Rather than using the clustering on the data domain, we first create a learned tabular embedding space. Using an external embedding space allows for unsupervised enrichment of tabular data by capturing complex, high-order relationships for clustering methods. While such models enable the creation of high-fidelity embeddings of heterogeneous data [28, 25, 12], these models also increase the training time and memory overhead significantly. We use TransTab [28] to create these embeddings by using the pre-activation logits of the `[CLS]` token after training the TransTab model on a table. We compare this with our default encoding process using one-hot encoding for categorical features and standardization for numerical ones. This latter method imposes negligible computational overhead to 'embed' the data, provided that the categorical features are not too sparsely represented. As such, for the FTP dataset, we exclude the 'category_d' column from the dataset used for clustering, as these are almost all unique. By doing so, we reduce the encoded representation considerably, as the one-hot encoded representation of such features yields extremely sparse data to the clustering process.

Table 6: Relational-dataset evaluation on AirBnb for Discriminator Measure (DM), Logistic Detection (LD) scores, and train duration T_{train}. 'Learned Spaces' indicate the embedding space used assigning cluster ID in step ① of our method. Scores are reported as mean$_{\pm\text{standard deviation}}$. For DM, lower is better; for LD, higher is better. Best scores are highlighted in boldface and second-best are underlined.

Metric			Learned Spaces	
		REaLTabFormer	TransTab	One-Hot
Parent	LD	52.63$_{\pm 0.00}$	78.07$_{\pm 0.00}$	**87.24$_{\pm 0.00}$**
	DM	0.57$_{\pm 0.05}$	0.21$_{\pm 0.00}$	**0.17$_{\pm 0.06}$**
Child	LD	42.22$_{\pm 0.00}$	56.31$_{\pm 0.00}$	**67.18$_{\pm 0.00}$**
	DM	0.44$_{\pm 0.05}$	0.35$_{\pm 0.00}$	**0.33$_{\pm 0.07}$**
	T_{emb}	-	2735	1
	T_{train}	2762	3237	3228
	T_{total}	**2762**	5972	3229

In Table 5 we compare the difference in performance between the different clustering methods on the Airbnb dataset. While both embedding improve over REaLTabFormer performance, the One-Hot data yields greater improvements. Moreover, it requires considerably less compute, only adding around one second to the training process. Although more embedding mechanisms could be considered, combined with the already strong performance of the baseline method, we leave this to future work to explore.

6 Conclusion

In this work, we study relational table synthesizers and narrow the performance gap of complementary methods, namely diffusion-based and transformer-based generative models. First, we examine the effect of contextual data augmentation on transformer-based multi-tabular data synthesis, inspired by the success of this technique in state-of-the-art diffusion-based models. Secondly, we show the positive impact of layer-sharing by first flattening the relation, providing ground for our novel layer-sharing approach.

By incorporating these insights into the training data and architecture, we demonstrated an average improvement in the LD score of 1.22× for the parent tables in the relational benchmarks and a reduction in the DM by 2.08×. This confirms that contextual augmentation and information reuse can considerably enhance fidelity with minor additional costs. Building on this foundation, we propose using layer-sharing for transformer architectures, in which we copy and freeze model weights from the trained parent model to the decoder in the sequence-to-sequence model. Our experiments on relational datasets show that this form of knowledge transfer across relational schemas yields an average improvement of 1.73× in LD and 1.17× in DM on child tables. Together, these complementary strategies surpass naive transformer baselines and rival state-of-the-art diffusion approaches' performance in data fidelity and utility. For future work, we are working to extend the proposed method towards m-n relational and multi-relational (3+) data.

GenAI Usage Disclosure

Herein, we confirm that generative AI tools have been used exclusively to assist with rewriting or performing grammar and spelling checks on text written by the authors. All ideas, experimental designs, analyses, and core paper contents were created solely by the authors. No portions of the research code, data generation, or original contributions were created or modified by GenAI. Although this work investigates GenAI *as a method* to generate relational tabular data, in this paper, as the objective of this paper is to compare and improve the gap between diffusion-based and transformer-based tabular generative models. As such, it is necessary to evaluate those GenAI models and present their qualitative results.

References

[1] André Bauer, Simon Trapp, Michael Stenger, Robert Leppich, Samuel Kounev, Mark Leznik, Kyle Chard, and Ian T. Foster. 2024. Comprehensive exploration of synthetic data generation: A survey. *CoRR*, abs/2401.02524. arXiv: 2401.02524. doi:10.48550/ARXIV.2401.02524.

[2] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. (1996). doi:10.24432/C5XW20.

[3] R. Bock. 2004. Magic gamma telescope. UCI Machine Learning Repository. (2004). doi:10.24432/C52C8B.

[4] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. 2023. Language models are realistic tabular data generators. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/pdf?id=cEygmQNOeI.

[5] Leo Breiman. 2001. Random forests. *Mach. Learn.*, 45, 1, 5–32. doi:10.1023/A:1010933404324.

[6] Leo Breiman, J. H. Friedman, Richard A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Wadsworth. ISBN: 0-534-98053-8.

[7] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, (Eds.) ACM, 785–794. doi:10.1145/2939672.2939785.

[8] Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55, 1, 119–139. doi:10.1006/JCSS.1997.1504.

[9] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

[10] Alok Gupta, Anna Montoya, Liz Sellier, Meghan O'Connell, and Wendy Kan. 2015. Airbnb new user bookings. (2015). https://kaggle.com/competitions/airbnb-recruiting-new-user-bookings.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, (Eds.) https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

[12] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar S. Karnin. 2020. Tabtransformer: tabular data modeling using contextual embeddings. *CoRR*, abs/2012.06678. https://arxiv.org/abs/2012.06678 arXiv: 2012.06678.

[13] Valter Hudovernik, Martin Jurkovic, and Erik Strumbelj. 2024. Benchmarking the fidelity and utility of synthetic relational data. *CoRR*, abs/2410.03411. arXiv: 2410.03411. doi:10.48550/ARXIV.2410.03411.

[14] Florian Knauer and Will Cukierski. 2015. Rossmann store sales. (2015). https://kaggle.com/competitions/rossmann-store-sales.

[15] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2023. Tabddpm: modelling tabular data with diffusion models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA* (Proceedings of Machine Learning Research). Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, (Eds.) Vol. 202. PMLR, 17564–17579. https://proceedings.mlr.press/v202/kotelnikov23a.html.

[16] Bei Li, Tong Zheng, Yi Jing, Chengbo Jiao, Tong Xiao, and Jingbo Zhu. 2022. Learning multiscale transformer models for sequence generation. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA* (Proceedings of Machine Learning Research). Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, (Eds.) Vol. 162. PMLR, 13225–13241. https://proceedings.mlr.press/v162/li22ac.html.

[17] Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. 2023. GOGGLE: generative modelling for tabular data by learning relational structure. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. https://openreview.net/pdf?id=fPVRcJqspu.

[18] Jan Motl and Oliver Schulte. 2024. The ctu prague relational learning repository: ftp dataset. https://relational.fel.cvut.cz/dataset/FTP. Accessed: 2025-05-14. (2024). arXiv: 1511.03086 [cs.LG].

[19] R. Kelley Pace and Ronald Barry. 1997. Sparse spatial autoregressions. *Statistics and Probability Letters*, 33, 291–297.

[20] Wei Pang, Masoumeh Shafieinejad, Lucy Liu, and Xi He. 2024. Clavaddpm: multi-relational data synthesis with cluster-guided diffusion models. *CoRR*, abs/2405.17724. arXiv: 2405.17724. doi:10.48550/ARXIV.2405.17724.

[21] C. Sakar and Yomi Kastro. 2018. Online shoppers purchasing intention dataset. UCI Machine Learning Repository. (2018). doi:10.24432/C5F88Q.

[22] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Trans. Neural Networks*, 20, 1, 61–80. doi:10.1109/TNN.2008.2005605.

[23] Ruxue Shi, Yili Wang, Mengnan Du, Xu Shen, and Xin Wang. 2025. A comprehensive survey of synthetic tabular data generation. (2025). https://arxiv.org/abs/2504.16506 arXiv: 2504.16506 [cs.LG].

[24] Aivin V. Solatorio and Olivier Dupriez. 2023. Realtabformer: generating realistic relational and tabular data using transformers. *CoRR*, abs/2302.02041. arXiv: 2302.02041. doi:10.48550/ARXIV.2302.02041.

[25] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. 2021. SAINT: improved neural networks for tabular data via row attention and contrastive pre-training. *CoRR*, abs/2106.01342. https://arxiv.org/abs/2106.01342 arXiv: 2106.01342.

[26] Mihaela Catalina Stoian, Eleonora Giunchiglia, and Thomas Lukasiewicz. 2025. A survey on tabular data generation: utility, alignment, fidelity, privacy, and beyond. *CoRR*, abs/2503.05954. arXiv: 2503.05954. doi:10.48550/ARXIV.2503.05954.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, (Eds.), 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[28] Zifeng Wang and Jimeng Sun. 2022. Transtab: learning transferable tabular transformers across tables. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, (Eds.) http://papers.nips.cc/paper%5C_files/paper/2022/hash/1377f76686d56439a2bd7a91859972f5-Abstract-Conference.html.

[29] William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. 1993. Breast cancer wisconsin (diagnostic). UCI Machine Learning Repository. (1993). doi:10.24432/C5DW2B.

[30] Yingce Xia, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin. 2019. Tied transformers: neural machine translation with shared encoder and decoder. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 5466–5473. doi:10.1609/AAAI.V33I01.33015466.

[31] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional GAN. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, (Eds.), 7333–7343. https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html.

[32] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional GAN. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, (Eds.), 7333–7343. https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html.

[33] Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. Iryna Gurevych and Yusuke Miyao, (Eds.) Association for Computational Linguistics, 46–55. doi:10.18653/V1/P18-1005.

[34] Biao Zhang, Deyi Xiong, and Jinsong Su. 2016. Cseq2seq: cyclic sequence-to-sequence learning. *arXiv preprint arXiv:1607.08725*.

[35] Long Zhou, Yuchen Liu, Jiajun Zhang, Chengqing Zong, and Guoping Huang. 2018. Language-independent representor for neural machine translation. *CoRR*, abs/1811.00258. http://arxiv.org/abs/1811.00258 arXiv: 1811.00258.

3

Background and Related Works

This chapter expands upon the topics introduced in the research paper in chapter 2. We begin by establishing the theoretical and practical foundations of synthetic data, examining the evolution of generative modelling approaches within this context, and conclude with a detailed analysis of current state-of-the-art methods that form the basis for our research contributions. The chapter is organized into three main sections.

Section 3.1 introduces synthetic data generation from both theoretical and practical perspectives. This section explains why synthetic data is essential and what makes relational data synthesis particularly challenging. **Section 3.2** examines the architectural foundations underlying modern methods for generating synthetic data. This section provides crucial background for understanding the architectural trade-offs between these approaches. It establishes the context for our subsequent analysis of their relative performance in relational settings in chapter 4. **Section 3.3** presents a review of existing approaches and state-of-the-art models.

3.1. Foundations and Applications

Synthetic data refers to artificially generated data that preserves the statistical properties and relationships of real data [7]. Unlike real-world data, synthetic data is created through computational processes rather than being collected from real-world events or measurements.

Applications and Motivation Synthetic data is in high demand in multiple domains where traditional data sharing faces significant challenges due to privacy concerns, regulatory constraints, or data scarcity issues [6]. The **healthcare sector** represents one of the most compelling use cases for synthetic data generation due to the highly sensitive nature of patient information and strict regulatory requirements, such as the GDPR in Europe and the American Health Insurance Portability and Accountability Act (HIPAA) [34] in the United States. Synthetic patient data enables researchers to develop and validate machine learning algorithms

without exposing actual patient records [69]. This method is especially useful in the context of small sample sizes, such as for rare diseases or underrepresented demographic groups [27].

In the **financial sector**, synthetic data is primarily used to support the development of fraud detection systems and regulatory compliance testing [4]. By generating synthetic transactional datasets, institutions can develop and evaluate systems and algorithms without compromising sensitive customer information. Credit scoring models also benefit from synthetic customer profiles that reflect a wide range of demographic and financial characteristics while preserving individual privacy. Finally, the use of synthetic financial data facilitates collaboration between financial institutions and academic researchers, enabling advancements in financial technology while upholding confidentiality standards.

Synthetic data is also increasingly used for **research and development** by both academic and industry research communities to promote reproducibility and enable data sharing across different institutions. By offering standardized benchmarks, synthetic datasets allow for the comparison of algorithmic approaches without requiring access to proprietary or sensitive data. This is particularly beneficial in interdisciplinary collaborations, where organizations may hold complementary datasets but are unable to share them directly due to privacy constraints or competitive interests. Moreover, synthetic data helps address the issue of research reproducibility by providing shareable datasets that support independent validation of findings.

Finally, **machine learning applications** also benefit from the use of synthetic data. Data augmentation is a key application area for synthetic data, particularly in situations involving imbalanced datasets or rare event detection [14, 24, 6]. Generating synthetic examples for under-represented classes can help mitigate class imbalance, a common challenge in real-world machine learning tasks. This approach is advantageous when acquiring additional real-world data is costly, time-consuming, or ethically constrained. In addition, synthetic data supports more comprehensive model evaluation by enabling testing across a broader range of scenarios that may not be sufficiently represented in the original training data.

Synthetic Data Fundamentals As highlighted in the previous section, synthetic data can be used to overcome many of the challenges relevant to working with data. In this thesis, our primary focus is on relational data stored in tabular format. Before exploring the challenges associated with both single-table tabular data and relational data, we define how the quality of synthetic data can be evaluated.

Assessment of Synthetic Data Quality The quality of synthetic data is typically assessed across three dimensions [1]. *Statistical fidelity* (fidelity) measures how well the synthetic data preserves the distribution characteristics of the original dataset. In other words, the fidelity of generated data measures how closely the synthetic data resembles the original real data. *Downstream utility* (utility) assesses whether synthetic data maintains sufficient quality for practical applications, such as training machine learning models or conducting statistical analyses. The data utility thus measures the usefulness of the synthetic data when used in a

downstream task. *Privacy preservation* evaluates the degree to which synthetic data protects against re-identification attacks and membership inference, ensuring that individual records cannot be traced back to real entities [67]. This thesis focuses on the data fidelity and utility measures, leaving privacy preservation assessment for future work.
These quality dimensions exist in tension with one another, creating what is commonly referred to as the *privacy-utility trade-off* [77]. Increased privacy protection through greater data transformation typically results in a reduction in statistical fidelity and downstream utility. Conversely, maintaining high utility often requires preserving fine-grained statistical relationships that may inadvertently leak sensitive information [67].

Synthetic Tabular Data Generating synthetic tabular data presents unique challenges that distinguish it from other synthetic data domains, such as images or text. Tabular datasets typically contain heterogeneous feature types, including categorical variables, continuous numerical values, ordinal features, and binary indicators, each requiring specialized handling during the generation process [74].
One key challenge of tabular data is the presence of mixed data types. Categorical variables, particularly those with varying cardinalities (the number of unique values per categorical variable), require distinct treatment from numerical features, and their interactions can introduce additional complexity into the modeling process. High-cardinality categories, such as user IDs or geographic codes. present unique difficulties due to sparse representation and a high risk of memorization [35]. Furthermore, tabular datasets often include heterogeneous feature distributions, where some features follow conventional distributions, while others exhibit heavy tails, multimodality, or complex interdependencies. Generative models must preserve these statistical properties and maintain logical coherence across features. Class imbalance and rare categories introduce an additional layer of complexity, particularly in domains such as fraud detection or healthcare, where positive instances are either scarce or overrepresented. Generative models must accurately capture these low-frequency events while avoiding *mode collapse*, where the model generates only a limited set of frequent patterns, and over-smoothing, where the outputs become overly averaged and lose meaningful variation across samples [76, 67]. Finally, tabular data often contains feature dependencies that reflect domain-specific rules or logical constraints—for example, age may correlate with education level, or specific combinations of features may be invalid. A good generative model must preserve these relationships without overfitting to the training data.

Synthetic Relational Data Compared to creating synthetic data for a single flat table, generating synthetic data for relational databases presents considerable difficulties, stemming from their composition of several interconnected tables. In the context of relational datasets, specifically those stored using relational database management systems (RDBMS), *keys* are attributes or sets of attributes that uniquely identify rows within a table or define relationships between tables [17]. Common types include *primary keys*, which ensure row uniqueness, and *foreign keys*, which enforce referential integrity between related tables. Because primary and

foreign key links create complex dependency structures that mirror actual relationships in the data, maintaining the integrity of these relationships is crucial in multi-table setups. This work focuses on datasets where one parent table is connected to one child table, a setting we refer to as relational. Datasets with more than two tables are referred to as multi-table datasets, which is out of the scope for this work but is covered by some recent methods [30, 54].

Additional Complexity of Relational Data Maintaining referential integrity is the key challenge in generating synthetic relational datasets. Primary keys must remain unique within each table, and foreign keys must accurately reference existing primary key values from their parent tables. Violations of these constraints create records that disrupt the logical structure and render the generated synthetic data invalid. To address this challenge, contemporary methods such as ClavaDDPM [54] explicitly enforce these constraints by generating parent tables before their children tables in a coordinated, schema-aware process. Accurately replicating cardinality relationships between tables presents another crucial aspect for generating realistic datasets. *One-to-many connections*, such as the relationship between customers and their orders, require managing the distribution of child rows associated with each parent to maintain consistency and realism. *Many-to-many relationships*, such as datasets of students that can enroll in multiple courses, are often implemented via junction tables and increase complexity by requiring consistency across multiple relational paths.

Time-based constraints add another layer of challenge, as many schemas require events to follow logical sequences, such as customer registration preceding order placement and payment records following order completion. Properly managing these causal sequences requires modeling more than just static referential links; it involves capturing business logic and domain-specific constraints. Finally, cross-table semantic constraints, such as ensuring that employee salaries fall within the defined ranges for their job or that student enrollment dates occur within valid academic periods, must be respected. Without these checks, synthetic datasets risk containing logically impossible or inconsistent scenarios. Effective methods combine schema- and constraint-aware design with generative modeling to ensure multi-table logical consistency.

3.2. Generative Modeling Approaches and Architectures

This section examines the architectural changes that have led to the development of the latest models for relational synthetic data. We examine the evolution from traditional statistical methods to current deep learning approaches, focusing on how each model addresses the challenges of tabular data. The section provides a detailed analysis of transformer and diffusion model structures, comparing their benefits and drawbacks for synthetic data output. This architectural understanding forms the basis for identifying specific areas where transformer models can be improved for relational data synthesis.

Evolution of Synthetic Data Generation Early synthetic data generation relied on traditional statistical methods. Parametric approaches used standard distributions, such as Gaus-

sian or copula models, to generate new samples [7]. These methods worked well for simple datasets but struggled with complex relationships and high-dimensional data. Non-parametric techniques, such as bootstrap sampling, offered more flexibility but could not generate truly novel examples beyond resampling existing data points. Machine learning brought new possibilities. Decision trees and Bayesian networks could model conditional dependencies between variables, enabling more sophisticated synthetic data generation. Bayesian networks proved particularly useful for tabular data as they could capture feature relationships explicitly. However, these approaches still faced limitations when dealing with complex non-linear patterns and high-dimensional feature spaces.

Deep learning transformed synthetic data generation through several key paradigms. Generative Adversarial Networks (GANs) [28] introduced adversarial training approaches for generating realistic data. Variational Autoencoders (VAEs) [42] established probabilistic frameworks for learning data representations. More recently, diffusion models have emerged as powerful alternatives with enhanced stability and data quality [36]. Transformer architectures, initially designed for natural language processing [70], have also been adapted for tabular data generation, bringing the power of attention mechanisms to synthetic data tasks.

Generative Adversarial Networks (GANs) GANs train two neural networks in competition: a generator creates synthetic samples while a discriminator tries to identify fake data [28]. The generator G learns to map random noise $z \sim p_z(z)$ to synthetic samples $G(z)$, aiming to produce data indistinguishable from the real distribution $p_{\text{data}}(x)$. Simultaneously, the discriminator D attempts to distinguish real samples $x \sim p_{\text{data}}$ from generated ones. Training proceeds through a minimax game with the value function being defined by Equation 3.1.

$$\min_G \max_D \; V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)} \left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)} \left[\log\left(1 - D(G(z))\right)\right] \tag{3.1}$$

Here, $D(x)$ estimates the likelihood that a sample x comes from the real data distribution, while $G(z)$ generates synthetic samples from random noise z. The generator learns to produce data that can deceive the discriminator, which in turn learns to better distinguish between real and fake data, creating an adversarial process that pushes both networks to improve. Figure 3.1 presents an overview of the GAN architecture, showing the computation of both the discriminator and generator loss and how they influence both models during training.

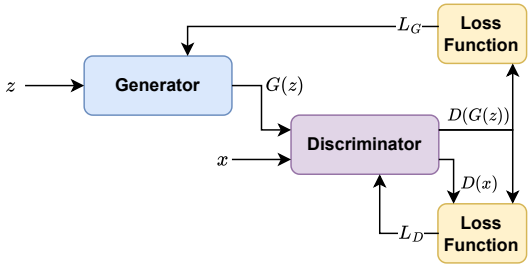


Figure 3.1: GAN architecture.

For tabular data, the GAN framework faces unique challenges. Mixed data types require special handling, as standard GAN architectures typically work with continuous values. CTGAN [74] addresses this by using mode-specific normalization for continuous features and categorical encodings for discrete variables. Training instability represents another significant challenge for tabular GANs. Mode collapse can cause the generator to produce limited diversity in synthetic samples. Class imbalance in the training data can lead to inadequate representation of minority classes in the generated samples. Various techniques have been proposed to address these issues, including modified loss functions and training procedures.

Variational Autoencoders (VAEs) VAEs combine neural networks with probabilistic modeling to learn latent representations of data [42]. An encoder maps inputs to a latent distribution, typically a Gaussian distribution, and a decoder reconstructs samples from this space, enabling the generation of new data using the Training a VAE involves maximizing the evidence lower bound (ELBO) on the data likelihood, which balances the accuracy of reconstruction and regularization of the latent space. For an input x, the ELBO is given by Equation 3.2.

$$\mathcal{L}_{\text{ELBO}}(x) = \mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - D_{\text{KL}}\left(q_\phi(z|x) \,\|\, p(z)\right) \tag{3.2}$$

Here, $q_\phi(z|x)$ is the encoder, or approximate posterior, $p_\theta(x|z)$ is the decoder, modeling the likelihood of the data given the latent variables, and $p(z)$ is the prior over the latent space, typically chosen as a standard normal distribution. The reconstruction term encourages the model to reproduce the input data accurately. In contrast, the Kullback–Leibler (KL) divergence term regularizes the latent space to remain close to the prior, which helps ensure smooth interpolation and meaningful structure in the generative process.

TVAE [74] adapts this framework for mixed-type tabular data by using a Bayesian Gaussian Mixture Model for continuous features and one-hot encoding for categorical features. VAEs offer more stable training than GANs, with reconstruction loss and KL divergence providing reliable learning signals, especially useful for small datasets. However, they can suffer from posterior collapse, where the model ignores the latent variables, resulting in smooth or unrealistic samples. Additionally, the Gaussian latent assumption may limit their expressiveness for complex tabular distributions.

Graph Neural Networks (GNNs) GNNs offer a novel approach to tabular data generation by representing features as nodes and their dependencies as edges [62]. These graphs can be constructed using statistical measures, such as correlation, or learned during training, sometimes incorporating domain knowledge to improve their structure. The core mechanism of GNNs is *message passing*, where each node iteratively aggregates information from its neighbors to update its representation. At layer $l+1$, the representation $h_v^{(l+1)}$ of node v is updated via Equation 3.3.

$$h_v^{(l+1)} = \sigma\left(\sum_{u\in\mathcal{N}(v)} f_\theta^{(l)}\left(h_v^{(l)}, h_u^{(l)}, e_{uv}\right)\right) \tag{3.3}$$

Here, $\mathcal{N}(v)$ denotes the neighbors of node v, $h_v^{(l+1)}$ is the node's embedding at layer l, e_{uv} represents edge features, $f_\theta^{(l)}$ is a learnable message function, and σ is a non-linear activation function.

Through iterative message passing, GNNs can capture complex, higher-order relationships between features, making them potentially well-suited for structured generative modeling tasks. However, their application to tabular data presents significant challenges. Unlike natural graph domains such as social networks, tabular data lacks inherent *graph topology*, the structural arrangement of connections between data points, requiring researchers to construct artificial graphs based on feature correlations or domain knowledge. This graph construction process is often non-trivial and domain-specific. Additionally, GNNs introduce computational overhead compared to traditional tabular methods and suffer from limited interpretability, as the learned feature interactions through message passing are challenging to explain or validate.

Denoising Diffusion Probabilistic Models (DDPMs) Diffusion models work by gradually adding noise to data during a forward process, then learning to reverse this process to generate new samples [36]. The forward diffusion process converts data to pure noise through a series of small steps. The training objective of DDPMs is to learn the reverse denoising process by minimizing the difference between the true noise and the predicted noise. The forward process adds Gaussian noise over T steps, as defined in Equation 3.4.

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I}) \tag{3.4}$$

To approximate the reverse process, the model uses a neural network $\epsilon_\theta(x_t, t)$ and minimizes the simplified loss shown in Equation 3.5.

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right] \tag{3.5}$$

Here, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is the true noise, x_t is the noisy version of x_0 at timestep t, and ϵ_θ is trained to denoise it. This loss function encourages the model to learn an accurate reverse process, enabling high-quality sample generation.

TabDDPM [44] adapts diffusion models specifically for tabular data. It handles mixed data types by applying different noise schedules to categorical and continuous features. The model learns to denoise tabular data while preserving feature relationships and distributions. Diffusion models generally exhibit more stable training than GANs and can produce high-quality samples, though they require more computational resources during generation.

Transformer Architectures Transformers use self-attention mechanisms to model relationships between sequence elements [70]. Each layer of a transformer employs a self-attention mechanism to update the representation of each input token, taking into account its relationships with all other features in the input. This allows the model to capture complex dependencies between features, regardless of their position or type. The core operation behind this is

called *scaled dot-product attention*, as defined in Equation 3.6.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{3.6}$$

Here, Q, K, and V are the query, key, and value, respectively. They are computed by applying learned linear projections to the input embeddings. Intuitively, the query Q represents what each token is trying to focus on, the key K represents the content of each token, and the value V contains the information to be aggregated. The attention mechanism works by comparing each query to all keys using a dot product, determining how much attention each token should pay to other tokens. The resulting attention weights are normalized with the softmax function and used to compute a weighted sum of the values V, yielding an updated representation for each token. To stabilize gradients during training, the dot products are scaled by a factor of $\sqrt{d_k}$, where d_k is the dimensionality of the keys.

There are three main types of transformer architectures: decoder-only transformers, encoder-only transformers, and encoder-decoder transformers. **Decoder-only transformers** such as GPT-2 [59] and Llama [68] generate outputs one token at a time by attending only to past inputs, which is why they are also referred to as *autoregressive* models. When predicting the value of a token at position t, these models prevent attending to information from token positions $t+1, t+2, \ldots$, by applying masking to these future attention weights. This type of self-attention is called *causal self-attention*. Decoder-only transformers are commonly used for text or data generation tasks. This is the primary architecture used in most generative models. **Encoder-only transformers** such as BERT [20], encode the entire input sequence simultaneously using *bidirectional self-attention*, capturing rich contextual representations. These are mainly used for classification, regression, or embedding tasks where no generation is required. **Encoder-decoder transformers** such as T5 [60] and BART [46], use the encoder to process the input and the decoder to generate an output sequence, attending to both the encoder's output and previously generated tokens. This setup is effective for tasks like translation, summarization, or conditional data generation.

Adapting transformers for tabular data requires tokenizing the samples of each table while maintaining the per-column information. GReaT [10] tokenizes tabular rows into textual representations of feature-value pairs, enabling pre-trained language models to generate synthetic tabular data without changing the tokenizer. The attention mechanism allows transformers to model complex feature dependencies. Each feature can attend to all other features in the table, potentially capturing long-range relationships that simpler models might miss. However, transformers also face challenges with tabular data, including the need for careful tokenization strategies and position encoding schemes that make sense for non-sequential tabular features.

Transformer vs. Diffusion: Comparative Analysis A key difference between transformer-based and diffusion-based models lies in **data preprocessing**. Diffusion-based models op-

erate exclusively in numerical feature spaces, requiring all categorical or textual features to be encoded before training. Commonly used encoding schemes include label encoding, one-hot encoding, frequency encoding, target encoding, or learning embeddings. Each encoding scheme carries a trade-off in terms of bias, dimensionality of the encoded data, and generalization performance [32, 51]. This makes data preprocessing for diffusion models time-consuming and prone to performance issues when selecting the incorrect encoding scheme. One-hot encoding, a standard encoding scheme within the field of machine learning [57], for example, yields poor results when used to train diffusion models because of the sparse and high-dimensional encoding space generated by one-hot encoding [63]. In contrast to diffusion models, transformer-based models require no data preprocessing due to their tokenized approach, which can handle different data types natively. This allows transformer-based models to be trained directly on the raw, unprocessed data, making them more flexible and often easier to deploy across multiple datasets [70, 21].

In terms of **computational cost**, transformers rely on the self-attention mechanism that computes attention weights between all token pairs in an input sequence. This leads to a computational and memory complexity of $O(n^2)$, where n is the sequence length[70]. Consequently, transformers can become inefficient for long or high-dimensional tabular data representations. On the other hand, diffusion models scale linearly with the input size, as each forward and reverse step operates independently over the entire feature vector, regardless of the number of features or tokens [36, 40]. This makes diffusion-based models more computationally efficient for datasets where tokenization of features would otherwise result in long sequences.

Another significant distinction lies in the **conditioning of features during generation**. Transformer-based models, due to their flexible attention mechanism, can condition generation on arbitrary subsets of the input. By masking or providing only selected tokens, the model can infer missing values or generate samples consistent with partial inputs, enabling powerful forms of data imputation, augmentation, and conditional generation without architectural changes [19]. This conditioning behavior effectively provides an empirical estimate of the distribution of missing or target tokens, based on the observed features. In contrast, diffusion models typically require architectural adaptations, such as concatenating conditional inputs, adding feature-specific embeddings, or designing conditional noise schedules, to support similar types of conditioning [64].

3.3. Existing Approaches

This section focuses on existing synthetic generative data models and their evolution from mainly GAN-based models to transformer- and diffusion-based models of recent years. A more detailed analysis is provided for REaLTabFormer and ClavaDDPM, which serve as representative examples of diffusion and transformer architectures for relational data generation. These models serve as the foundation for our research, and understanding their capabilities and limitations is essential for identifying opportunities for architectural improvements.

GAN-Based Methods MedGAN [16] represents one of the earliest applications of GANs in the context of tabular data. In combination with an auto-encoder, MedGAN learns the distribution of discrete variables of electronic health records while maintaining acceptable levels of privacy risk. Table-GAN [55] extended the DCGAN [58] framework specifically for general tabular data generation, moving beyond the domain-specific application of MedGAN. This model introduced architectural changes to handle mixed data types often found in tabular data, including both continuous and categorical variables. TGAN [73] incorporated domain-specific knowledge about tabular data structures and introduced techniques for better handling of categorical variables through specialized encoding schemes. The model demonstrated improved fidelity in preserving statistical properties of the original data compared to its predecessors. CTGAN [74] introduced conditional generators that could be trained to generate samples based on specific categorical conditions, significantly improving the quality of generated data for imbalanced datasets while also being able to handle multi-model non-Gaussian data, and severe imbalance of categorical columns. CTGAN also incorporated mode-specific normalization techniques to effectively handle skewed continuous distributions. CTAB-GAN [78] advanced conditional GANs for tabular data by effectively modeling diverse data types, including a mix of continuous and categorical variables, and addressing data imbalance and long-tail distributions. CTAB-GAN achieved this by introducing classification loss, information loss, and generator loss into the conditional GAN framework, along with a conditional vector designed for mixed data types and skewed distributions. CTAB-GAN+ [79] further improved upon CTAB-GAN by incorporating an auxiliary classifier/regressor for additional supervision, enabling more efficient modeling of continuous, categorical, and mixed variables through a novel data encoding scheme. CTAB-GAN+ enhanced GAN training stability and effectiveness using the Wasserstein GAN with Gradient Penalty (Was+GP) and included information, downstream, and generator losses.

VAE-Based Methods VAEs offer another powerful approach for synthetic data generation by learning a latent representation of the data distribution. TVAE [74], developed by the same authors as CTGAN, adapted a VAE for mixed-type tabular data generation. Similar to CTGAN, it leveraged mode-specific normalization for continuous columns and conditional generation based on categorical columns to handle class imbalance. Graph-VAE [49] extended the VAE framework by integrating GNNs to generate realistic synthetic relational data. This method was shown to accurately preserve the structures of real datasets, even for large datasets with advanced data types. VAE-GMM [2] proposed a novel VAE-based model that integrates a Bayesian Gaussian Mixture Model within the VAE architecture. This approach overcomes the limitations of assuming a strictly Gaussian latent space, common in traditional VAEs like TVAE, by leveraging the inherent distribution approximation capabilities of GMMs. It also offers enhanced flexibility through the use of various differentiable distributions for individual features, allowing it to effectively handle both continuous and discrete data types.

Hybrid Methods TabSyn [76] is a hybrid model that combines two transformers in an autoencoder setup to convert raw tabular inputs into a continuous latent space. A score-based diffusion model operates within the learned latent space to capture and generate the distribution of the training dataset.
GOGGLE [47] (Generative mOdellinG with Graph LEarning) is a framework for generating synthetic tabular data by learning both the relationships between features and how those features influence one another. It builds on a VAE, where the decoder is a Message Passing Neural Network (MPNN). This MPNN uses a learnable, weighted graph that represents the inferred structure of dependencies between features.

Diffusion-Based Methods DDPMs, initially successful in image synthesis, have recently been adapted for tabular data, demonstrating competitive performance. However, applying diffusion models to tabular data with mixed data types is not straightforward. Diffusion models natively operate in continuous state spaces because their core mechanism involves gradually adding Gaussian noise to data and then learning to reverse this process, which inherently requires continuous-valued representations. This diffusion and denoising process relies on differentiable transitions and smooth probability distributions, features not present in discrete state spaces, where data changes occur in jumps rather than smoothly. In discrete spaces, such as text or categorical labels, there's no natural way to define or add Gaussian noise, nor to compute gradients through sampling steps, making it difficult for standard diffusion models to handle such data directly. Recent work [5] has provided the foundations for applying diffusion models to discrete state spaces, enabling their use with tabular data.

TabDDPM [44] introduced a simple yet effective DDPM design specifically for tabular data synthesis, capable of handling mixed data types. TabDDPM addresses heterogeneity by using two separate diffusion processes: a DDPM with Gaussian noise for numerical columns and a multinomial diffusion model for categorical columns. TabDDPM has consistently demonstrated superior performance over GAN-based and VAE-based alternatives in generating tabular data.
STaSy [41] applies score-based generative modeling to tabular data synthesis, addressing the training instability that arises from tabular data's complex multi-modal distributions through a self-paced learning strategy that progressively trains on samples from easy to hard, followed by fine-tuning. This approach demonstrates superior performance compared to existing GAN-based and other generative methods for tabular data. CoDi [45] proposed using two diffusion models, one for numerical and one for categorical columns, that are inter-conditioned on each other to model the joint distribution. It also adopted contrastive learning methods to bind the two diffusion processes further. RelDDPM [48] introduced a novel framework for controllable tabular data synthesis using diffusion models, capable of guiding the generative process to fulfill different conditions. RelDDPM specifically utilizes an unconditional DDPM tailored for tabular data, handling categorical attributes by incorporating a multinomial diffusion process, similar to TabDDPM. Conditioning is achieved through the use of controller modules that can steer the generation of synthetic samples at inference time without requiring retraining of the model. RelDDPM is suitable for multi-table datasets; however, since it requires initial condi-

tioning to generate samples and cannot be used for complete table synthesis as is, we exclude it from our benchmark experiments.

Diffusion-Based Methods: ClavaDDPM ClavaDDPM (Cluster Latent Variable Guided Denoising Diffusion Probabilistic Model) [54], also referred to as Clava, is designed for the synthesis of multi-relational (multi-table) datasets. This setting extends beyond the standard two-table relational setting covered in this thesis, allowing for both deeper relations and multiple related parent tables. In doing so, ClavaDDPM addresses the challenges related to scalability and capturing long-range dependencies across tables.

The fundamental innovation of ClavaDDPM lies in its use of clustering labels as intermediaries to model relationships between tables, with a particular focus on foreign key constraints. Rather than directly modeling the conditional distribution $p(g_j|yj)$ where g_j represents a foreign key group and y_j is the parent row, ClavaDDPM introduces latent variables c to achieve conditional independence: $g_j \perp y_j|c$. This allows the data modeling problem to be reformulated as shown in Equation 3.7.

$$p(g_j, y_j) = \sum_c p(g_j|c)p(y_j|c)p(c) = \sum_c p(g_j|c)p(y_j, c) \tag{3.7}$$

The conditioning space of the parent row y can be both noisy and of high dimensionality, which can lead to poorly learned conditional distributions and worsen the quality of generated samples. By conditioning on a cluster label, the conditioning space complexity is reduced while maintaining the ability to capture inter-table correlations.

ClavaDDPM operates using three distinct phases: **Phase 1: Latent learning and table augmentation**. First, the latent variables c are learned on the joint space $(X; Y)$, with X the child table data and Y the parent table data, using Gaussian Mixture Models (GMMs) [18]. Then relation-aware clustering is applied in the weighted joint space $H = (X; \lambda Y)$ where λ controls the importance balance of child versus parent features. Using the learned latent variables, the data is then augmented to create $T_Y = (Y; C)$. **Phase 2: Training**. Using the augmented parent table T_Y from phase 1, the diffusion model $p_\theta(y, c)$ is trained, after which the child diffusion model $p_\phi(x)$ is trained on the child table data. Then, a classifier $p_\psi(c|x)$ is trained to determine which cluster label a child row belongs to based on its features. Lastly, the foreign key group size s, where s represents the number of child rows referring to each parent row, is estimated using $p(s|c)$. **Phase 3: Synthesis**. The trained parent model p_θ can now be used to generate the synthetic parent table $\tilde{T}_Y$, augmented with the synthetic cluster labels $\tilde{C}$, using $\tilde{T}_Y = (\tilde{Y}; \tilde{C}) \sim p_\theta(\cdot, \cdot)$. Then, for each synthetic latent variable $\tilde{c}_j$, the sample group size $\tilde{s}_j \sim p(\cdot|\tilde{c}_j)$ is sampled to determine the number of child samples to be generated. Finally, the child rows are generated using the classifier-guided sampling: $\tilde{x}_j^i \sim p_{\phi,\psi}(\cdot|\tilde{c}_j)$, where $\tilde{x}_j^i$ denotes the i-th synthetic child row belonging to parent j. $p_{\phi,\psi}(\cdot|\tilde{c}_j)$ denotes the combination of using the child table diffusion model p_ϕ and the classifier p_ψ to generate child table samples conditioned on the cluster label.

Transformer-Based Methods Recent advancements in Large Language Models (LLMs) have created new opportunities for generating synthetic tabular data by harnessing the strong generative capabilities and proficiency in handling textual representations of data of LLMs. GReaT [10] introduced the first approach for synthesizing realistic, heterogeneous tabular data using an autoregressive generative LLM. It builds on a decoder-only GPT-2 architecture [59] as its backbone. To apply language modeling to tabular data, GReaT converts rows into textual sequences by combining feature names with their corresponding values. During training, it also permutes feature orders to allow for arbitrary conditioning at inference time. TabMT [31] introduced a novel Masked Transformer (MT) architecture for generating synthetic tabular data. TabMT is specifically designed to handle heterogeneous feature types and missing values natively, using enhanced masking strategies to guide the generative process. TabMT has achieved state-of-the-art performance across a diverse range of tabular datasets, from small to larger-scale datasets, and offers improved trade-offs between data utility and privacy.

Transformer-Based Methods: REaLTabFormer REaLTabFormer (Realistic Relational and Tabular Transformer) [65], referred to as RTF, is a transformer-based method designed for generating both single-table tabular data and relational datasets. RTF utilizes an auto-regressive decoder-only GPT-2 [59] transformer model to treat tabular data generation as a language modeling task, enabling the synthesis of realistic synthetic data. Each table row is treated as a sequence with potential dependencies across column values, similar to sentences in natural language processing. The conditional probability for x_{ij}, the value in row i and column j, is modelled as the conditional probability of column j, given the previous columns as shown in Equation 3.8.

$$x_{ij} \sim P(X|x_{i1}, x_{i2}, \ldots, x_{ij-1}) \tag{3.8}$$

For the relational setting, RTF extends this approach to model child table generation conditioned on the parent table context as shown in Equation 3.9.

$$x^n_{ij} \sim P(X|o^1_i, \cdots, x^n_{i1}, x^n_{i2}, \cdots, x^n_{ij-1}, C_k) \tag{3.9}$$

This conditioning thus happens on three types of information; $o^1_i, \cdots$ represents all previously generated child observations for the parent to capture the inter-observation dependencies, $x^n_{i1}, x^n_{i2}, \cdots, x^n_{ij-1}$ represents the previous columns in the current child observation being generated to capture the intra-observation dependencies and finally C_k is the encoded context from the parent table observation which captures the parent-child relationship. This way, RTF generates each table cell by considering both the sequential structure within rows and the relational structure across tables.

For relational data generation, RTF uses a three-phase approach: **Phase 1: Parent table training**. RTF uses a GPT-2 model with a language modeling head, as depicted by the third blue box in 3.2, to model the tabular parent table data. This yields a trained parent table model. **Phase 2: Child table training**. In the second phase, RTF implements a sequence-

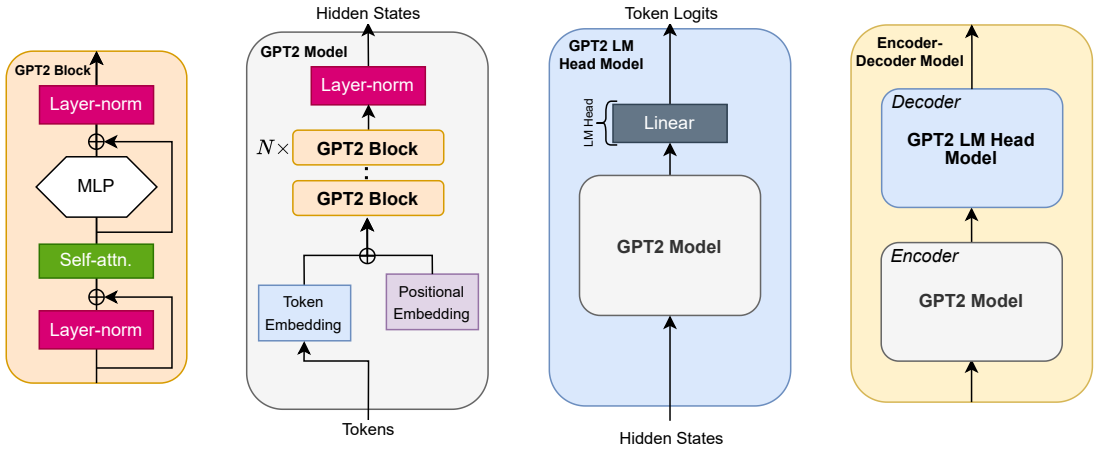


Figure 3.2: GPT-2 block, GPT-2 model, GPT-2 model with language modeling head and a sequence-to-sequence model.

to-sequence (Seq2Seq) architecture, as shown by the yellow rightmost box in 3.2, to learn both the child table distribution and the parent-child relationships. The encoder is taken as the pre-trained parent model from phase 1 without the language modeling head, and the decoder is a new GPT-2 model with a language modeling head. By default, RTF freezes all the weights in the encoder since those have already been trained. The decoder is capable of generating arbitrary-length sequences of child observations after training. **Phase 3: Synthesis**. First, the parent model is used to generate parent table samples, then these samples are used as conditional input for the generation of related child samples by the Seq2Seq model. All related child observations for the same parent table sample get concatenated into a sequence $s_i = [o_i^1, o_i^2, \cdots, o_i^n]$.

RTF employs an entirely text-based strategy for tokenizing data, including numerical values. Numerical values are tokenized per column. First, all numbers get rounded to a specified precision, then padding is applied using both leading and trailing zeros to make the number of characters per column entry equal, after which the resulting string is partitioned into fixed-length tokens that are then used to create column-specific vocabularies.

Another technique introduced is *target masking*, which acts as a regularization technique by randomly replacing target tokens with special mask tokens during training. This prevents the model from copying training data during data generation, forcing it to generalize and thereby reducing data copying risks. At inference time, the generation of the special mask token is disabled to prevent the production of invalid output samples.

To prevent overfitting, the Q_δ statistic is combined with boostrapping to detect overfitting without hold-out data: $Q_\delta = \frac{1}{N}\sum_q (p_q - q)$, where p_q represents the proportion of synthetic samples below the q-th quantile of the real data distribution. This is computed every 5 training epochs. Whenever the synthetic data is more similar to the sampled subset compared to the real data in 2 or more consecutive evaluations, training is stopped.

4

Experiments

This chapter presents a comprehensive experimental evaluation of our proposed synthetic data generation method against established baselines across multiple datasets and evaluation metrics. We compare our approach, which combines layer sharing and contextual cues, against ClavaDDPM and the RealTabFormer (RTF) baseline, alongside ablation studies that isolate the individual contributions of layer sharing and contextual cues.

Our evaluation framework utilizes data fidelity measures and distributional similarity metrics to assess the quality of synthetic data from multiple perspectives. The experimental design follows a systematic approach to validate three key questions: whether our combined method outperforms existing baselines in generating realistic tabular data, how each component (layer sharing and contextual cues) contributes to overall performance, and whether the improvements are consistent across different evaluation methodologies. We employ LD and DM metrics for data fidelity assessment, complemented by distributional similarity measures (Wasserstein distance and Maximum Mean Discrepancy) and classification-based overlap assessment (AUC), applied to UMAP projections to provide a comprehensive view of synthetic data quality. All experiments are conducted on standardized train-test splits to ensure fair comparison.

The following sections detail the experimental setup, introduce the datasets used, present comparative results across different datasets, and provide an analysis of the key findings that validate the effectiveness of our approach.

4.1. Experimental Setup

This section outlines the experimental setup used to conduct the experiments presented in this chapter and the research paper included in chapter 2.

Hardware All experiments were conducted on a workstation equipped with an AMD Ryzen 7950X CPU, 64 GB of DDR5 RAM, and an NVIDIA RTX 4090 featuring 24 GB of VRAM.

Please note that due to a mismatch in the required CUDA drivers, all results for the GOGGLE baseline were obtained by running on the CPU instead of the GPU, resulting in longer training times.

Metrics To assess the **data utility** of synthetic data, we use the Machine Learning Efficacy (MLE) [75] for single-table evaluation, which measures how effectively machine learning models perform when trained on synthetic data and tested on real samples. To mitigate bias from model selection, we employ a diverse set of five machine learning models per dataset: Decision Tree (DT) [13], Random Forest (RF) [12], Gradient Boosting [26], AdaBoost [25], and XGBoosted Random Forest [15]. For each trained model, we generate 5 distinct samples with the same length as the test dataset. Performance scores are aggregated, and the arithmetic mean and standard deviation are reported.

Data Preprocessing Since the metrics introduced in the previous paragraph all solely work on numerical data, encoding schemes are needed to evaluate generated samples. For the Breast Cancer, California Housing and Magic datasets no encoding is needed since all features are fully numerical. For the Adult Income and Shoppers dataset, one-hot encoding is used to convert raw samples generated from models like RTF or samples in possibly different encodings, such as those from ClavaDDPM, into the same space. For the relational datasets, we encode the foreign key columns using *incremental label encoding* and pick the encoding for the other columns depending on the dataset. The Airbnb dataset contains many categorical features; therefore, applying one-hot encoding would result in a dataset that is too high-dimensional. To avoid the ordered data space of label encoders, we encode the AirBnB dataset using an adjusted version of frequency encoding, which we label as *centered frequency encoding*. This type of encoding eliminates the possibility of two distinct categories being mapped to the same encoded value, which can occur naturally with standard frequency encoding. Let X be a categorical variable with categories $c_1, c_2, \ldots, c_k$ and corresponding relative frequencies $f(c_1), f(c_2), \ldots, f(c_k)$, sorted in descending order of frequency. The centered frequency encoding enc(c_i) for category c_i can be computed as defined in Equation 4.1.

$$\text{enc}(c_i) = \left(\sum_{j=1}^{i} f(c_j) \right) - \frac{f(c_i)}{2} \tag{4.1}$$

The Rossmann dataset is encoded with one-hot encoding for both the parent and child tables. For the FTP dataset, the ‘sessions’ parent table, which contains information on user online browser sessions, only has one categorical value, representing the gender of the user; we encode this using a simple label encoder. The ‘products‘ child table of the FTP dataset holds data on what products users have been viewing during their browser sessions and consists of a session ID linking the viewing to a user session in the parent table. The remaining five features are categorical, grouping the viewed product in specific subcategories. One of these categories, ‘category_d’, has $|\text{category\_d}| = 36,092$ while the total number of samples in the

product table is 66,491. Applying one-hot encoding to this table is thus not feasible, and using centered frequency encoding would map all values to be close to each other, thereby losing much of the context. As such, this table is encoded using incremental label encoding.

A simple experiment running ClavaDDPM on the Adult dataset using three encoding schemes shows that centered frequency encoding can bring benefits in both data fidelity and utility scores for numerical methods. These results can be found in Table 4.1

Table 4.1: MLE and DM scores for training ClavaDDPM on the Adult Income dataset using different encoding schemes; incremental label encoding (Label Enc.), centered frequency encoding (Freq. Enc.), and one-hot encoding (One-Hot Enc.).

Metric	Original	Label Enc.	Freq. Enc.	One-Hot Enc.
MLE	$0.10_{\pm 0.00}$	$\underline{0.11}_{\pm 0.00}$	$\mathbf{0.10}_{\pm \mathbf{0.00}}$	$0.15_{\pm 0.00}$
DM	$0.00_{\pm 0.00}$	$\underline{0.08}_{\pm 0.03}$	$\mathbf{0.06}_{\pm \mathbf{0.03}}$	$0.40_{\pm 0.06}$
T_{train}	-	**355**	$\underline{401}$	435

4.2. Datasets

Table 4.2: Overview of datasets used, with names of the Parent and *Child* Tables for relational datasets. In the 'Task' column Bin. denotes Binary, Class. denotes Classification, Multi. denotes Multi-class and Reg. denotes regression.

	Dataset	**Table**	**#Rows**	**#Columns**		**Task**
				Cat.	Num.	
Single-table	Breast Cancer Wisconsin [72]		569	1	30	Bin. Class.
	California Housing [53]		20,640	0	9	Reg.
	Adult Income [8]		45,222	9	6	Bin. Class.
	Magic Gamma Telescope [9]		19,020	1	9	Bin. Class.
	Shoppers [61]		12,330	7	9	Bin. Class.
Relational	AirBnB User Bookings [33]	Users	10,000	13	3	Multi. Class.
		Sessions	191,025	5	1	
	Rossmann Store Sales [43]	Stores	1,115	3	7	Reg.
		Sales	68,015	3	6	
	FTP [52]	Sessions	15,000	1	3	Bin. Class.
		Products	33,455	5	1	

Table 4.2 gives an overview of all datasets used as benchmarks within this thesis. The following paragraphs provide a brief introduction to each dataset, beginning with the five single-table datasets and concluding with three relational datasets, each comprising one parent and one child table.

Single-Table Datasets

The **Breast Cancer Wisconsin** (Breast Cancer) [72] dataset contains diagnostic measurements from breast mass samples, where the task involves classifying tumors as malignant or benign based on computed features from digitized images. This binary classification problem involves 30 numerical features derived from cell characteristics, including radius, texture, and smoothness. The number of samples is limited to 569.

The **California Housing** (Cali. Housing) [53] dataset presents a regression challenge involving median house values of homes in California. The dataset incorporates geographical and demographic variables, including median income, housing age, average rooms per household, and population density, making it suitable for evaluating models on continuous target prediction.

The **Adult Income** (Adult) [8] dataset, derived from the 1994 US Census, poses a binary classification task to predict whether an individual's annual income exceeds $50K. This dataset combines categorical features such as education level, marital status, and occupation with numerical attributes like age and hours worked per week, providing a realistic scenario with mixed data types.

The **MAGIC Gamma Telescope** (Magic) [9] dataset originates from high-energy physics experiments aimed at distinguishing gamma-ray signals from background noise in telescope observations. The binary classification task relies on features extracted from image parameters, offering a domain-specific challenge with only numerical inputs.

The **Online Shoppers Purchasing Intention** (Shoppers) [61] dataset captures user behavior during e-commerce website sessions to predict purchase completion. Features include session duration metrics, page visit counts, traffic sources, and temporal information, representing a modern application area with practical commercial relevance.

Relational Datasets

To evaluate the performance of the benchmark and proposed generative relational models, we use three datasets, each with one parent and one child table.

The **AirBnB New User Bookings** (AirBnB) [33] dataset involves predicting the first booking destination for new users, incorporating user demographics, session logs, and destination information.

The **Rossmann Store Sales** (Rossmann) [43] dataset focuses on forecasting daily sales for a chain of stores, utilizing historical sales data, store attributes, and external factors like promotions and holidays.

The **FTP** [52] dataset contains user browsing sessions in the parent table with corresponding product viewings in the child table. Initially, the task corresponding to the dataset was predicting the gender of a user based on all their product viewings.

4.3. Data Representation Experiments

Our experiments regarding different data representations are two-fold; first, we evaluate whether the performance of RTF is affected when we create a relational dataset by splitting a single-table dataset into two separate tables. The second experiment we perform evaluates the effect of *data denormalization*, in which we join the parent and child tables of the relational datasets to yield a single joint table; more details can be found in the research paper in chapter 2. The results of this experiment correspond to Table 2 of the research paper and are presented in corrected form in Table A.1.

Training RTF on Split California Housing Dataset

We selected the California Housing dataset as a representative single-table dataset with a substantial number of samples and features. As the primary key, we take the 'MedInc' and 'AveOccup' columns, which represent the median household income and the average number of occupants per household, respectively.

Table 4.3: MLE and DM scores for training RTF model on the split California Housing dataset. 'Full' denotes an RTF model with 12 layers in the GPT-2 models instead of the default of 6 layers.

Metric	Original	RTF	Left to Right		Right to Left	
			RTF	RTF Full	RTF	RTF Full
MLE	$0.35_{\pm 0.09}$	**$0.36_{\pm 0.08}$**	$5.89_{\pm 0.24}$	$3.23_{\pm 0.21}$	$0.57_{\pm 0.05}$	$0.56_{\pm 0.07}$
DM	$0.01_{\pm 0.00}$	**$0.16_{\pm 0.04}$**	$0.89_{\pm 0.02}$	$0.84_{\pm 0.03}$	$0.74_{\pm 0.05}$	$0.68_{\pm 0.06}$
T_{train}	-	652	**611**	1381	779	1432

Experiment Summary: Training RTF model on the split California Housing dataset. Training is performed starting with the left side of the table as the parent table and the right side as the child table (Left to Right) and the other way around (Right to Left).

Key Takeaway: Starting training from the right table performs significantly better compared to starting with the left side. Performance is still degraded compared to training on a single table.

4.4. Layer Sharing

This section covers experiments related to our proposed layer-sharing technique, as introduced in the research paper. We investigate what layers to share and how many layers to use.

What Layers to Share?

Our layer-sharing technique can be applied to any subset of the GPT-2 layers. When a layer is selected to be shared between the encoder and decoder in the Seq2Seq model, its weights

get frozen during training on the child table. Any layer that is not being shared does not get frozen and will be trained from a random initialization.

Table 4.4: Evaluation of sharing different layers with evaluation on Discriminator Measure (DM), Logistic Detection (LD) scores, and train duration. Scores are reported as mean$_{\pm \text{standard deviation}}$. For DM, lower is better; for LD, higher is better. Best scores are highlighted in **boldface** and second-best are underlined.

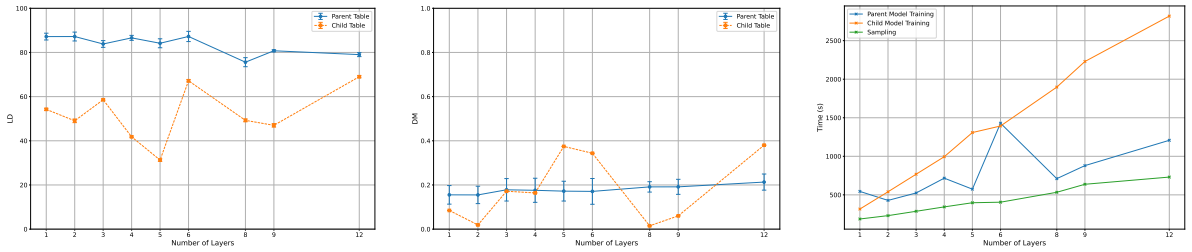
		Metric	Original	Shared Layers				
				RTF	First 2	Middle 2	Last 2	All
AirBnB	Parent	LD	97.82$_{\pm 1.66}$	**52.63$_{\pm 1.55}$**	50.85$_{\pm 2.51}$	52.57$_{\pm 2.45}$	52.46$_{\pm 2.19}$	51.55$_{\pm 1.36}$
		DM	0.02$_{\pm 0.01}$	0.57$_{\pm 0.05}$	**0.56$_{\pm 0.04}$**	**0.56$_{\pm 0.05}$**	**0.56$_{\pm 0.05}$**	0.57$_{\pm 0.04}$
	Child	LD	91.23$_{\pm 1.05}$	42.22$_{\pm 1.65}$	28.73$_{\pm 0.57}$	28.47$_{\pm 0.78}$	20.61$_{\pm 0.32}$	**71.85$_{\pm 0.75}$**
		DM	0.05$_{\pm 0.02}$	0.44$_{\pm 0.05}$	0.53$_{\pm 0.04}$	0.55$_{\pm 0.06}$	0.64$_{\pm 0.05}$	**0.35$_{\pm 0.07}$**
		T_{train}	-	2762	2535	2321	**2286**	4347

Experiment Summary: We compare layer sharing by only sharing a subset or all layers of the GPT-2 model between the encoder and decoder in the sequence-to-sequence model. When layer sharing is applied, the shared layers are frozen, while the remaining layers remain unfrozen.

Key Takeaway: Layer sharing yields no performance improvement and even degrades performance when not applied to all layers.

Number of Layers

After establishing the importance of applying layer sharing to all layers in the GPT-2 model, we aim to find out what a good number of layers to use is. By default, inspired by GReaT and RTF, we use the distilled configuration of GPT-2 which has 6 layers with an embedding size of 768.



(a) Logistic Detection **(b)** Discriminator Measure **(c)** Train and Sampling Times

Figure 4.1: Layer sharing experiment to evaluate the impact of the number of layers when using layer sharing.

Table 4.5: Comparison of the number of layers used by our model. The experiments under 'Number of Layers' include both layer sharing and contextual cues.

	Metric	Original	RTF	Number of Layers							
				1	2	3	4	5	6	9	12
Parent	LD	97.82±1.66	52.63±1.55	87.18±1.54	87.20±2.01	83.83±1.52	86.60±1.12	84.16±0.00	**87.24±2.31**	80.76±0.00	79.02±0.00
	DM	0.02±0.01	0.57±0.05	0.16±0.04	**0.15±0.04**	0.18±0.05	0.18±0.05	0.17±0.01	0.17±0.06	0.19±0.00	0.21±0.00
Child	LD	91.23±1.05	42.22±1.65	54.23±0.48	49.05±0.83	58.58±0.34	41.80±0.35	31.27±0.00	67.18±0.53	47.00±0.00	**69.00±0.00**
	DM	0.05±0.02	0.44±0.05	0.46±0.06	0.51±0.09	0.43±0.10	0.52±0.07	0.58±0.00	**0.33±0.07**	0.46±0.00	0.35±0.00
	T_{train}	-	2762	**1048**	1198	1579	2055	2280	3228	3746	4757

Experiment Summary: Comparison of using different numbers of layers for the GPT-2 encoder and decoder when applying layer sharing and using contextual cues. RTF represents the baseline RTF model without any changes made.

Key Takeaway: The default number of six transformer layers yields the best performance for both the parent and child tables across LD and DM scores. Using more than six layers appears to impact performance and training time negatively. A smaller number of layers, specifically one or three, does yield surprisingly good LD scores for the parent table at the cost of lower LD and DM scores for the child table.

4.5. Contextual Cue

Building on our experiments using contextual cues, as introduced in the research paper, we present two new experiments summarized in Table 4.6. To obtain the contextual cues, our data needs to be in a fully numerical space. Since the selected relational datasets contain mixed data types, we require encoding or learning embeddings of the raw data. In the research paper, we've shown that using learned embeddings from TransTab [71] yields worse performance compared to a simple one-hot encoding, while also requiring significantly more training time.

To optimize the contextual cues, we compare different clustering methods, inspired by ClavaDDPM [54], and evaluate their performance in both one-hot encoded and label-encoded feature spaces.

Data Encoding & Clustering Method

ClavaDDPM introduced four different clustering techniques: using a Gaussian Mixture Model (GMM), using a Bayesian Gaussian Mixture Model with variational inference (Variational) for soft probabilistic cluster assignments, using k-Means clustering (k-Means) for hard partitioning, and using a hybrid approach that combines GMM with K-means++ initialization for improved convergence (Both).

Table 4.6: Performance comparison across different data encoding methods and clustering techniques.

		Metric	Original	RTF	One-Hot Encoding				Label Encoding			
					GMM	Variational	k-Means	Both	GMM	Variational	k-Means	Both
AirBnB	Parent	LD	97.82$_{\pm 1.66}$	52.63$_{\pm 1.55}$	85.00$_{\pm 2.27}$	**87.08$_{\pm 2.91}$**	84.46$_{\pm 1.36}$	82.83$_{\pm 0.87}$	75.11$_{\pm 1.32}$	78.24$_{\pm 1.14}$	86.37$_{\pm 2.13}$	79.22$_{\pm 1.71}$
		DM	0.02$_{\pm 0.01}$	0.57$_{\pm 0.05}$	0.19$_{\pm 0.05}$	**0.16$_{\pm 0.05}$**	0.24$_{\pm 0.09}$	0.17$_{\pm 0.05}$	0.32$_{\pm 0.08}$	0.26$_{\pm 0.07}$	0.17$_{\pm 0.04}$	0.31$_{\pm 0.11}$
	Child	LD	91.23$_{\pm 1.05}$	42.22$_{\pm 1.65}$	**66.64$_{\pm 0.64}$**	56.20$_{\pm 1.20}$	59.17$_{\pm 0.66}$	62.07$_{\pm 0.47}$	53.66$_{\pm 0.68}$	43.63$_{\pm 0.59}$	57.62$_{\pm 0.72}$	61.91$_{\pm 0.88}$
		DM	0.05$_{\pm 0.02}$	0.44$_{\pm 0.05}$	**0.28$_{\pm 0.05}$**	0.37$_{\pm 0.04}$	0.33$_{\pm 0.05}$	0.34$_{\pm 0.05}$	0.38$_{\pm 0.04}$	0.44$_{\pm 0.04}$	0.35$_{\pm 0.04}$	0.32$_{\pm 0.05}$
		T_{train}	-	2762	2747	2756	4514	**2738**	3094	2804	2742	4522

Experiment Summary: Comparing performance for generating contextual cues using different feature spaces and clustering techniques.

Key Takeaway: One-hot encoding provides the best performance improvement using variational clustering, albeit at the cost of worse child table performance compared to GMM-based clustering in the same feature space. Label encoding is outperformed by one-hot encoding across all techniques, except for k-means clustering, where better LD and DM performance is achieved for the parent table.

4.6. Extended Analysis & Visualization

To gain further insights into the effects of layer sharing and the use of contextual cues, we present extended visualizations and statistical analyses in this section. This builds upon the experiments introduced in the research paper.

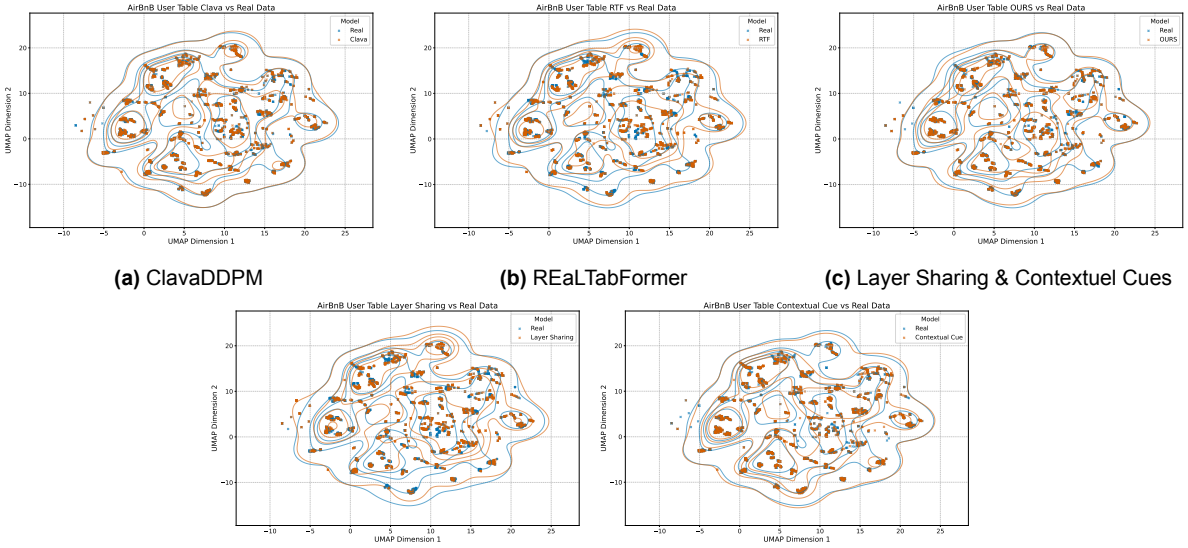
Working with tabular data provides a challenge in how to visualize the data, plotting the distribution of each column gives some insights into the performance of each column, but lacks the structure and relations across the features. We use a UMAP [50] projection to convert the higher-dimensional tabular data to a 2D feature space. UMAP is a non-linear dimensionality reduction technique that projects high-dimensional data into lower-dimensional spaces while preserving both local and global structure of the data. Unlike linear methods like PCA, UMAP can capture complex, non-linear relationships in the data, and unlike t-SNE, it better preserves global structure and is more computationally efficient. For synthetic data evaluation, UMAP is particularly valuable because it reveals the underlying manifold structure of both real and generated datasets, allowing visual assessment of whether synthetic data occupies the same regions of the data space as real data. By comparing UMAP embeddings of real versus synthetic data, we can determine whether the generative model successfully captures the actual data distribution's clustering patterns, density variations, and overall geometric structure—aspects that are difficult to assess through traditional statistical summaries alone.

We train a UMAP model on the real AirBnB test data and the synthetic data generated by ClavaDDPM, the RTF baseline, our method, which comprises both layer sharing and using contextual cues, and the ablations of only using contextual cues or layer sharing. The learned 2D embeddings are visualized through scatter plots that display the spatial distribution of real versus synthetic samples, with different colors distinguishing between the two. To better reveal

density patterns and distributional overlap, we overlay Kernel Density Estimation (KDE) [56] contours on the scatter plots, which highlight regions of high data concentration and provide a more precise assessment of whether synthetic samples occupy similar areas in the embedding space as the real data. In these visualizations, we look for better overlap between real and synthetic data distributions, where fewer isolated blue points, representing real data, indicate superior coverage by the synthetic model and thus higher data quality.

UMAP Visualization AirBnB User Table

For the user table, we take one synthetic sample per model and the whole test dataset as real data to train a UMAP model.



(a) ClavaDDPM **(b)** REaLTabFormer **(c)** Layer Sharing & Contextuel Cues

(d) Layer Sharing **(e)** Contextual Cues

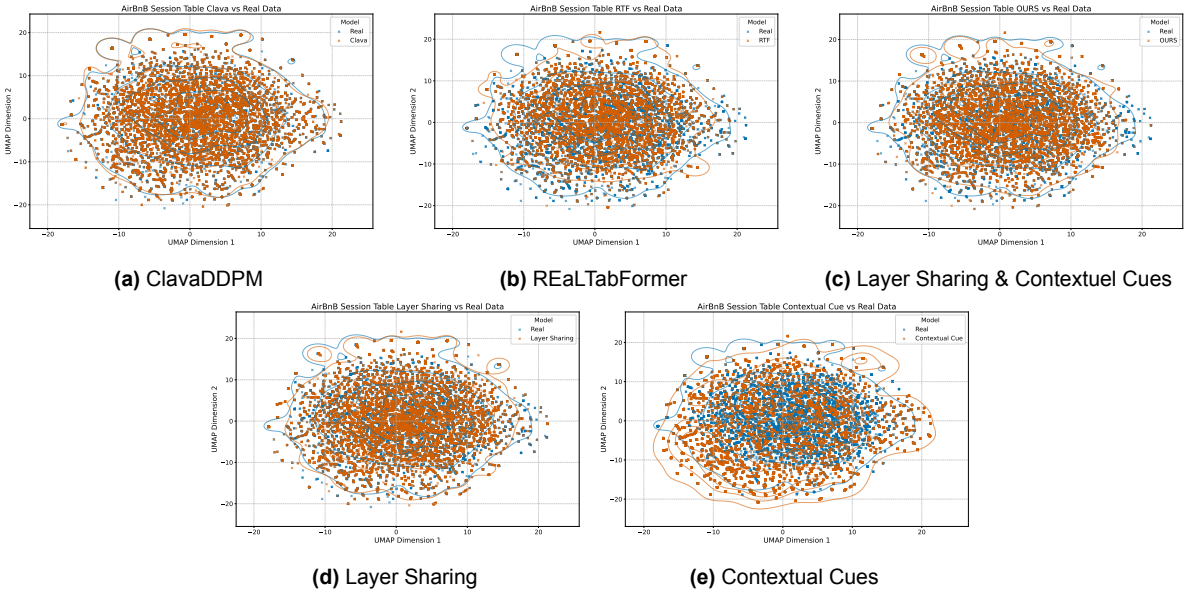
Figure 4.2: UMAP projections of the AirBnB User table and the generated samples by ClavaDDPM, REaLTabFormer, our model with layer sharing only, our model with contextual cues only, and finally our model with both layer sharing and contextual cues

Experiment Summary: Comparing the UMAP projections of different generative models on the AirBnB user table to see the impact of our proposed methods on the distribution of the generated data.

Key Takeaway: Contextual cues improve the generated data compared to the baseline RTF model, as reflected by better data overlap in the UMAP plots for Figure 4.2c and Figure 4.2e.

UMAP Visualization AirBnB Session Table

Given the high number of samples in the AirBnB session test dataset (39,167 versus 2,000 in the user table), we take one sample per model from which we randomly sample about a third, 13,055.



(a) ClavaDDPM **(b)** REaLTabFormer **(c)** Layer Sharing & Contextuel Cues

(d) Layer Sharing **(e)** Contextual Cues

Figure 4.3: UMAP projections of the AirBnB Session table and the generated samples by ClavaDDPM, REaLTabFormer, our model with layer sharing only, our model with contextual cues only, and finally our model with both layer sharing and contextual cues

Experiment Summary: Comparing the UMAP projections of different generative models on the AirBnB session table to see the impact of our proposed methods on the distribution of the generated data.

Key Takeaway: Layer sharing improves the generated data compared to the baseline RTF model, as reflected by better data overlap in the UMAP plots for Figure 4.3c and Figure 4.3e. Contextual cues yield no performance improvement on their own for the session table.

UMAP Statistical Analysis

After training the UMAP model, we obtain embeddings of each synthetic sample. In addition to the visual comparison provided above, we also conduct a statistical analysis to assess how well the generated distributions represent the actual data, with results shown in Table 4.7. For distribution similarity, we use the Wasserstein distance, also referred to as the Earth Mover's Distance, which quantifies the minimum cost to transform one probability distribution into another, measuring both location and shape differences. Lower values indicate better overlap, with 0 representing identical distributions [3]. Also, we use the Maximum Mean Discrepancy (MMD) [29]. MMD is a kernel-based test that compares distributions by measuring distances between their mean embeddings, capturing higher-order statistical dependencies. Lower values indicate better similarity, with 0 representing identical distributions. As a third statistic, we use the Area Under the ROC Curve (AUC) [11], which is a classification performance metric that measures a classifier's ability to distinguish between two classes, ranging from 0.5 (equal to random guessing) to 1.0 (perfect data separation). For synthetic data evaluation, we train

a random forest classifier to distinguish between real and synthetic data, with the AUC quantifying how easily the datasets can be separated. Values close to 0.5 indicate good overlap and indistinguishable distributions, while values near 1.0 suggest easily separable data.

Table 4.7: Statistics derived from the UMAP-generated feature space for the Airbnb dataset. Wass. represents the Wasserstein distance, MMD the Maximum Mean Discrepancy, and AUC the Area Under the Curve. For Wass. and MMD, lower is better, while for AUC, a score closer to 0.50 is best. Scores are presented over one run.

	Metric	RTF	Clava	OURS	Ablation* (subsection 5.3 paper)	
					Layer Share	Context ID
Parent	Wass.	0.72	**0.22**	0.27	1.05	0.32
	MMD	0.0033	**0.0010**	0.0016	0.0038	0.0015
	AUC	0.67	**0.53**	0.62	0.70	0.56
Child	Wass.	0.56	**0.13**	0.38	0.43	2.34
	MMD	0.0019	**0.0002**	0.0011	0.0011	0.0035
	AUC	0.75	**0.54**	0.70	0.69	0.87

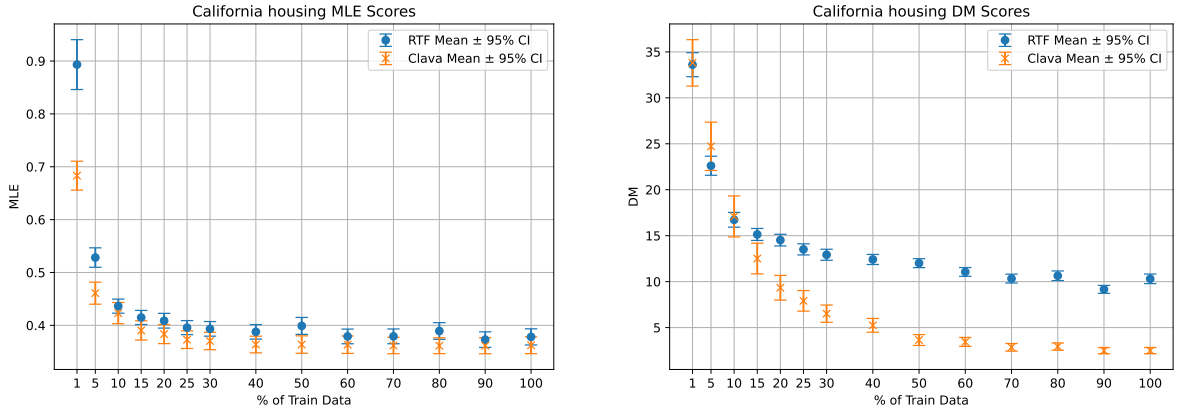
Table 4.7 shows that using contextual cues yields better data for the parent table, as indicated by both lower Wasserstein distance and MMD value, as well as an AUC score closer to 0.50, compared to the baseline RTF model. Furthermore, layer sharing improves the performance of the child table.

4.7. Other Experiments

This section covers additional experiments that provided no grounds for performance improvements but could be helpful for the reader in understanding the capabilities of transformer-based relational models.

Training on Partial Datasets

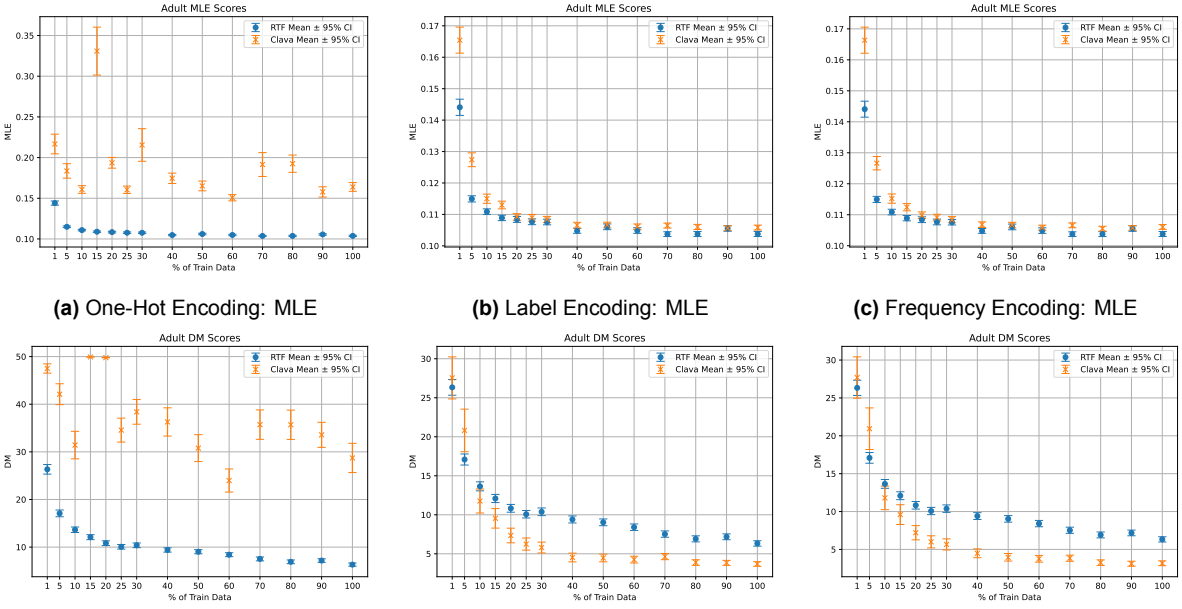
One of the things we investigated was whether transformer-based models could outperform diffusion-based models when both are trained on subsets of the training data. We test this by randomly sampling five different subsets of all single-table datasets, except for the Breast Cancer Wisconsin dataset, which, due to its limited number of samples, was excluded from the experiment. We then train both ClavaDDPM and the baseline RTF model on these subsets. We take samples ranging in size from 1% to 100% of the training data.



(a) Machine Learning Efficacy

(b) Discriminator Measure

Figure 4.4: California housing dataset: scores for training ClavaDDPM and REaLTabFormer on a subset of the training data.



(a) One-Hot Encoding: MLE

(b) Label Encoding: MLE

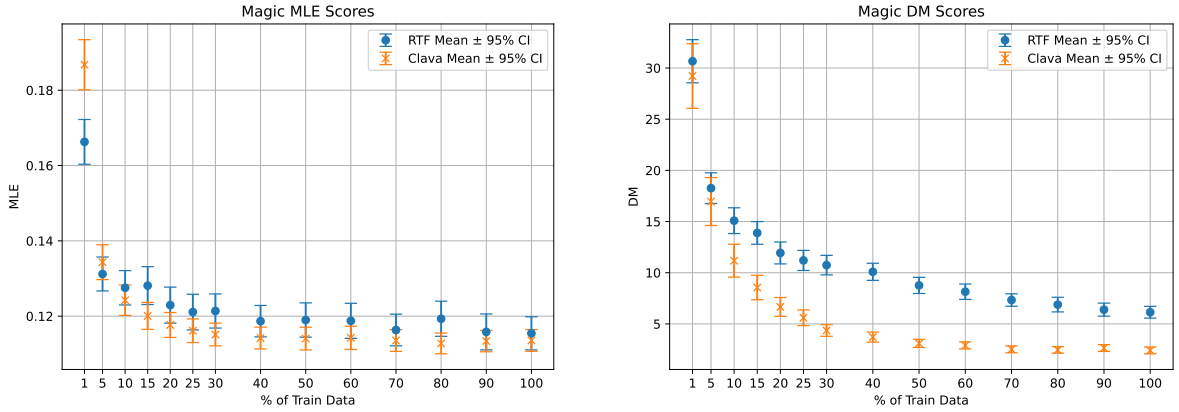
(c) Frequency Encoding: MLE

(d) One-Hot Encoding: DM

(e) Label Encoding: DM

(f) Frequency Encoding: DM

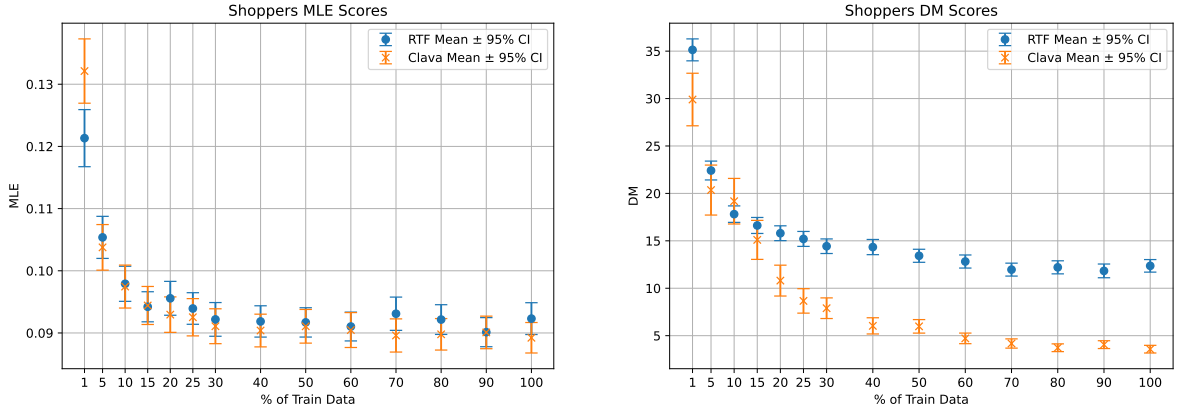
Figure 4.5: Adult income dataset: scores for training ClavaDDPM and REaLTabFormer on a subset of the training data. Different data encoding methods have been used for creating the dataset used by ClavaDDPM.



(a) Machine Learning Efficacy

(b) Discriminator Measure

Figure 4.6: Magic dataset: scores for training ClavaDDPM and REaLTabFormer on a subset of the training data.



(a) Machine Learning Efficacy

(b) Discriminator Measure

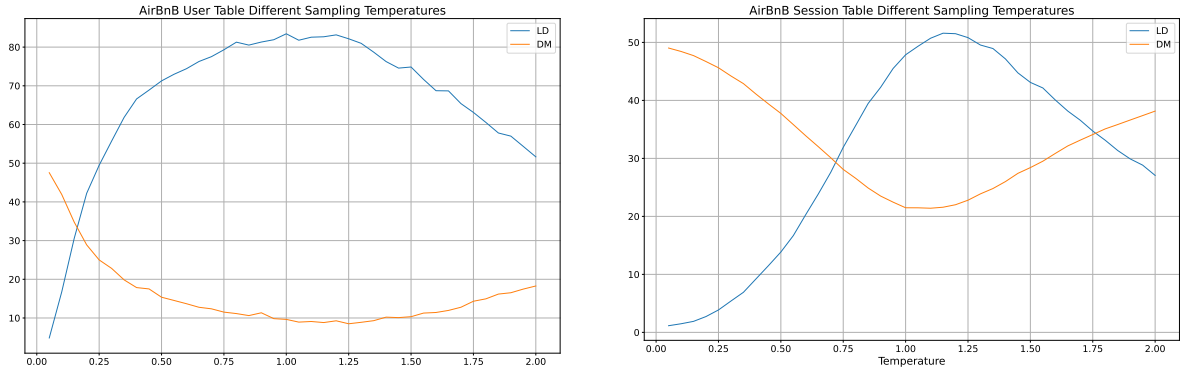
Figure 4.7: Shoppers dataset: scores for training ClavaDDPM and REaLTabFormer on a subset of the training data.

Experiment Summary: Sample a fraction of the training data across four different single-table datasets and train both ClavaDDPM and RTF on the sampled subsets. Per fraction, we perform five distinct training runs.

Key Takeaway: ClavaDDPM matches or outperforms RTF on almost all datasets, except for tiny fractions of the training data, such as 1% as can be seen in Figure 4.5a and Figure 4.5d. Both models seem to converge to near-optimal performance when around 15-25% of the training data is used. The performance of ClavaDDPM appears unstable when trained on the one-hot encoded Adult dataset, which can be attributed to the sparse feature space.

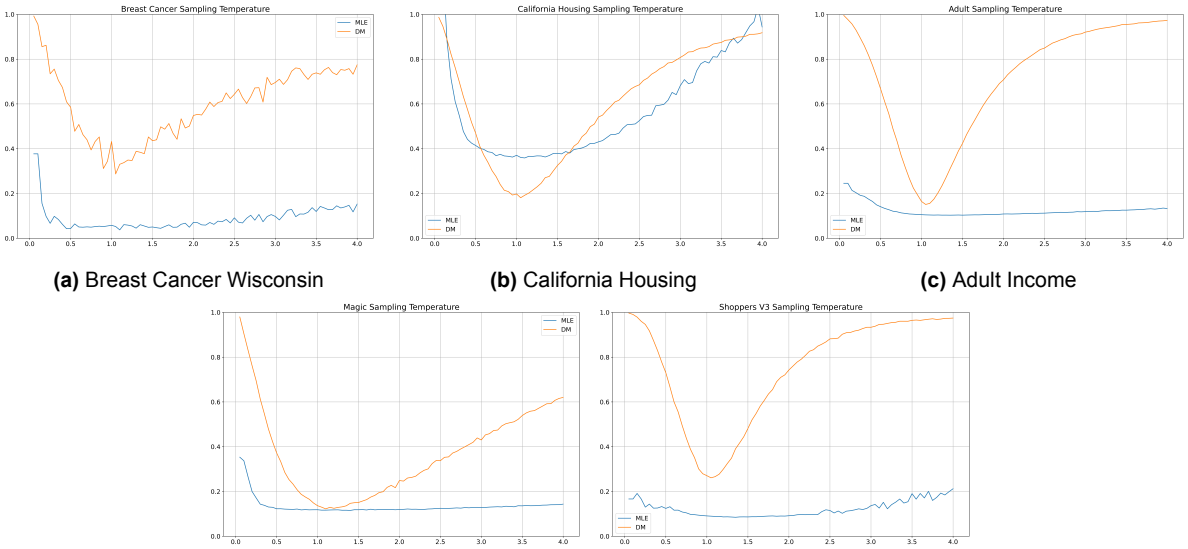
Tuning Temperature Parameter During Sampling

In this experiment, we investigate the effects of varying the temperature parameter when sampling from the trained baseline RTF model. The temperature parameter, commonly denoted as T, is a scalar that controls the randomness of sampling from transformer models. Lower temperatures ($T < 1$) make the model more deterministic, favoring high-probability tokens, while higher temperatures ($T > 1$) increase diversity by flattening the probability distribution, allowing lower-probability tokens to be sampled more frequently. By default, as is common for transformer models, RTF uses a temperature value of $T = 1.0$. Temperature is widely used to balance between repetition and creativity in generative models [37]. We run the experiment on all five single-table datasets and the AirBnB dataset.



(a) AirBnB User (Parent) Table **(b)** AirBnB Sessions (Child) Table

Figure 4.8: Results of sampling from a trained RTF model with different values for the temperature (x-axis) and the corresponding LD and DM metric results on the y-axis. For LD, higher is better, while for DM, lower is better.



(a) Breast Cancer Wisconsin **(b)** California Housing **(c)** Adult Income

(d) Magic **(e)** Shoppers

Figure 4.9: Changing the temperature parameter during sampling after training a REaLTabFormer model on single table datasets.

Experiment Summary: Varying the temperature hyperparameter T for a trained RTF model on different datasets.

Key Takeaway: Values below 1.0 yield worse performance in general, while a slightly higher temperature of up to 1.10 does improve performance on some datasets. Since the performance improvements are minor, we stick with the default setting of 1.0.

Different Decoder Architecture: Llama

One of the benefits of using transformers is that changing between architectures is straightforward, thanks to APIs like the Hugging Face Transformers library. In this experiment, we replace the GPT-2 decoder used by RTF with an LLaMA-based decoder and train the model on the Magic dataset. Hyperparameter tuning is applied to both the number of layers (L) and the size of the hidden states (H). Results are summarized in Table 4.8.

Table 4.8: Comparing the performance of ClavaDDPM, RTF, and RTF using the Llama decoder on the Magic dataset. '2L 4096H' denotes the usage of a Llama model with 2 layers and a hidden state size of 4096. 'Num. Params.' denotes the total number of model parameters.

	Clava	RTF	2L 4096H	2L 128H	1L 4096H	1L 2048H	1L 1024H	1L 512H	1L 256H	1L 128H	1L 64H
MLE	**0.11**$_{\pm 0.02}$	**0.11**$_{\pm 0.01}$	0.12$_{\pm 0.01}$	0.12$_{\pm 0.01}$	0.13$_{\pm 0.02}$	0.13$_{\pm 0.02}$	0.12$_{\pm 0.01}$	0.12$_{\pm 0.01}$	0.12$_{\pm 0.01}$	0.12$_{\pm 0.01}$	0.12$_{\pm 0.01}$
DM	**0.04**$_{\pm 0.03}$	0.11$_{\pm 0.03}$	0.27$_{\pm 0.07}$	0.05$_{\pm 0.02}$	0.19$_{\pm 0.05}$	0.16$_{\pm 0.04}$	0.14$_{\pm 0.04}$	0.10$_{\pm 0.03}$	0.06$_{\pm 0.03}$	0.05$_{\pm 0.02}$	0.08$_{\pm 0.04}$
Train Time (s)	**374**	1701	1738	1310	1211	583	430	539	1013	951	986
Num. Params.	-	43,775,232	409,677,824	8,739,200	207,294,464	86,870,016	39,240,704	18,571,776	9,023,744	4,446,336	2,206,784

Experiment Summary: Performance comparison of training an RTF model with a GPT-2 decoder versus an Llama decoder. We vary the number of layers and the size of the hidden states of the Llama decoder.

Key Takeaway: Using a Llama decoder can achieve MLE scores comparable to those of ClavaDDPM and RTF, while improving upon RTF in terms of DM scores and approaching those of ClavaDDPM. These results can be obtained using significantly fewer parameters.

Negative Results

While the core methods in this work produced promising results, we also explored several alternative approaches that ultimately did not yield improvements or introduced new challenges. We briefly summarize the most notable of these below to inform future work and help others avoid similar pitfalls.

Initially, we aimed to compute contextual cues based on learned spaces using custom models. To this end, we attempted to develop *Autoencoders* and *embedding models*, both of which proved challenging to stabilize during training and ultimately did not work. Should one be interested in obtaining embeddings, we strongly advise using existing models such as TransTab [71], SAINT [66], or TabTransformer [38].

During the development of our layer sharing method, we explored various options to utilize layer sharing without freezing the layers in the decoder of the Seq2Seq model after copying them from the encoder to the decoder; however, this approach did not yield positive results.

Hyperparameter tuning has consumed a significant portion of time during research, which can be beneficial for some parameters. However, we advise spending more time examining and understanding the generated data before proceeding in this direction.

5

Conclusion

This thesis evaluates the performance of transformer-based methods in comparison with their diffusion-based counterparts in both the single-table and relational settings. Identifying an existing performance gap between the two, this work addresses three key research questions aimed at closing this gap.

5.1. Discussion

Regarding **RQ1** on how relational data representation affects generative model performance, our experiments revealed that denormalizing datasets negatively impacted transformer-based model performance. This finding emphasizes the importance of preserving the original relational structure rather than flattening relationships, as the hierarchical organization inherent in multi-table schemas provides crucial information for effective generation.

For **RQ2** concerning the role of layer sharing in closing the performance gap between transformer-based and diffusion-based models, our layer-sharing mechanism demonstrated substantial improvements. By copying and freezing weights from trained parent models to child table decoders, we achieve average improvements of $1.80\times$ in LD and 2.10 reduction in DM scores for child tables in the relational setting. This approach effectively leverages the learned representations from the parent table, bringing transformer performance closer to that of state-of-the-art diffusion methods.

Addressing **RQ3** on how relational cue representation affects generation quality, our contextual data augmentation strategy proved highly effective. Incorporating contextual information during training, inspired by successful diffusion-based implementations, yielded average improvements of $1.23\times$ in LD scores for parent tables and $2.10\times$ reduction in DM metrics. The UMAP-based distributional analysis further validated these findings, showing superior overlap with real data distributions across Wasserstein distance, Maximum Mean Discrepancy, and

classification-based separability metrics.

The combination of these techniques demonstrates that architectural enhancements and data representation strategies can significantly improve transformer-based relational data generation while maintaining computational efficiency. Averaging over both parent and child tables, we achieve an average increase in LD score of $1.52\times$ and an average decrease in DM score of $1.94\times$

5.2. Future Work

Several promising directions and extensions emerge from this research for future investigation. First, extending our approach to handle deeper relational datasets beyond the current parent-child structure would enable the generation of more complex datasets. This includes supporting grandparent-parent-child relationships and many-to-many connections, which are common in real-world databases but present additional challenges for maintaining referential integrity across multiple relational paths.

Second, improving the tokenization strategy for numerical values represents a significant opportunity for enhancement. Current approaches treat numerical data similarly to categorical information, potentially losing important distributional properties. Developing specialized tokenization schemes that better preserve numerical relationships and distributions could further improve generation quality.

Second, investigating hybrid architectures that combine the strengths of diffusion and transformer models, similar to recent work like TabSyn [76], offers potential for achieving superior performance. Such approaches could leverage the distributional modeling capabilities of diffusion processes while maintaining the sequential generation advantages of transformer architectures.

Third, exploring multidimensional contextual cues could provide richer relational representations while also allowing for the introduction of another loss term, which would punish the generation of child samples that are far from their parent samples in the embedding space of the contextual cues.

Lastly, as also mentioned by the authors of the RTF paper [65], it can be worthwhile to look into swapping out the GPT-2 model for another transformer model. Results on one single-table dataset are discussed in chapter 4.

References

[1] Ahmed M. Alaa et al. “How Faithful is your Synthetic Data? Sample-level Metrics for Evaluating and Auditing Generative Models”. In: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 290–306. URL: https://proceedings.mlr.press/v162/alaa22a.html.

[2] Patricia A. Apellániz, Juan Parras, and Santiago Zazo. “An Improved Tabular Data Generator with VAE-GMM Integration”. In: *32nd European Signal Processing Conference, EUSIPCO 2024, Lyon, France, August 26-30, 2024*. IEEE, 2024, pp. 1886–1890. URL: https://ieeexplore.ieee.org/document/10715230.

[3] Martín Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein Generative Adversarial Networks”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 214–223. URL: http://proceedings.mlr.press/v70/arjovsky17a.html.

[4] Samuel A. Assefa et al. “Generating synthetic data in finance: opportunities, challenges and pitfalls”. In: *ICAIF ’20: The First ACM International Conference on AI in Finance, New York, NY, USA, October 15-16, 2020*. Ed. by Tucker Balch. ACM, 2020, 44:1–44:8. DOI: 10.1145/3383455.3422554. URL: https://doi.org/10.1145/3383455.3422554.

[5] Jacob Austin et al. “Structured Denoising Diffusion Models in Discrete State-Spaces”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc’Aurelio Ranzato et al. 2021, pp. 17981–17993. URL: https://proceedings.neurips.cc/paper/2021/hash/958c530554f78bcd8e97125b70e6973d-Abstract.html.

[6] Aayush Bansal, Rewa Sharma, and Mamta Kathuria. “A Systematic Review on Data Scarcity Problem in Deep Learning: Solution and Applications”. In: *ACM Comput. Surv.* 54.10s (2022), 208:1–208:29. DOI: 10.1145/3502287. URL: https://doi.org/10.1145/3502287.

[7] André Bauer et al. “Comprehensive Exploration of Synthetic Data Generation: A Survey”. In: *CoRR* abs/2401.02524 (2024). DOI: 10.48550/ARXIV.2401.02524. arXiv: 2401.02524. URL: https://doi.org/10.48550/arXiv.2401.02524.

[8] Barry Becker and Ronny Kohavi. *Adult*. UCI Machine Learning Repository. 1996. DOI: 10.24432/C5XW20. URL: https://doi.org/10.24432/C5XW20.

[9] R. Bock. *MAGIC Gamma Telescope*. UCI Machine Learning Repository. 2004. DOI: 10.24432/C52C8B. URL: https://doi.org/10.24432/C52C8B.

[10] Vadim Borisov et al. “Language Models are Realistic Tabular Data Generators”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/pdf?id=cEygmQNOeI.

[11] Ali Borji. “Pros and cons of GAN evaluation measures: New developments”. In: *Comput. Vis. Image Underst.* 215 (2022), p. 103329. DOI: 10.1016/J.CVIU.2021.103329. URL: https://doi.org/10.1016/j.cviu.2021.103329.

[12] Leo Breiman. “Random Forests”. In: *Mach. Learn.* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[13] Leo Breiman et al. *Classification and Regression Trees*. Wadsworth, 1984. ISBN: 0-534-98053-8.

[14] Nitesh V. Chawla et al. “SMOTE: Synthetic Minority Over-sampling Technique”. In: *J. Artif. Intell. Res.* 16 (2002), pp. 321–357. DOI: 10.1613/JAIR.953. URL: https://doi.org/10.1613/jair.953.

[15] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Ed. by Balaji Krishnapuram et al. ACM, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

[16] Edward Choi et al. “Generating Multi-label Discrete Electronic Health Records using Generative Adversarial Networks”. In: *CoRR* abs/1703.06490 (2017). arXiv: 1703.06490. URL: http://arxiv.org/abs/1703.06490.

[17] E. F. Codd. “A Relational Model of Data for Large Shared Data Banks”. In: *Commun. ACM* 13.6 (1970), pp. 377–387. DOI: 10.1145/362384.362685. URL: https://doi.org/10.1145/362384.362685.

[18] Arthur P Dempster, Nan M Laird, and Donald B Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the royal statistical society: series B (methodological)* 39.1 (1977), pp. 1–22.

[19] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

[20] Jacob Devlin et al. “BERT: Pre□training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of NAACL-HLT* (2019), pp. 4171–4186. URL: https://arxiv.org/abs/1810.04805.

[21] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[22] Yuntao Du and Ninghui Li. “Systematic Assessment of Tabular Data Synthesis Algorithms”. In: *arXiv preprint arXiv:2402.06806* (2024).

[23] European Parliament and Council of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council*. of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). OJ L 119, 4.5.2016, p. 1–88, May 4, 2016. URL: https://data.europa.eu/eli/reg/2016/679/oj (visited on 04/13/2023).

[24] Alberto Fernández et al. *Learning from Imbalanced Data Sets*. Springer, 2018. ISBN: 978-3-319-98073-7. DOI: 10.1007/978-3-319-98074-4. URL: https://doi.org/10.1007/978-3-319-98074-4.

[25] Yoav Freund and Robert E. Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”. In: *J. Comput. Syst. Sci.* 55.1 (1997), pp. 119–139. DOI: 10.1006/JCSS.1997.1504. URL: https://doi.org/10.1006/jcss.1997.1504.

[26] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[27] Andre Goncalves et al. “Generation and evaluation of synthetic patient data”. In: *BMC medical research methodology* 20 (2020), pp. 1–40.

[28] Ian J. Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani et al. 2014, pp. 2672–2680. URL: https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

[29] Arthur Gretton et al. “A Kernel Two-Sample Test”. In: *J. Mach. Learn. Res.* 13 (2012), pp. 723–773. DOI: 10.5555/2503308.2188410. URL: https://dl.acm.org/doi/10.5555/2503308.2188410.

[30] Mohamed Gueye, Yazid Attabi, and Maxime Dumas. “Row Conditional-TGAN for Generating Synthetic Relational Databases”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*. IEEE, 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10096001. URL: https://doi.org/10.1109/ICASSP49357.2023.10096001.

[31] Manbir Gulati and Paul F. Roysdon. “TabMT: Generating tabular data with masked transformers”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/90debc7cedb5cac83145fc8d18378dc5-Abstract-Conference.html.

[32] Cheng Guo and Felix Berkhahn. “Entity Embeddings of Categorical Variables”. In: *CoRR* abs/1604.06737 (2016). arXiv: 1604.06737. URL: http://arxiv.org/abs/1604.06737.

[33] Alok Gupta et al. *Airbnb New User Bookings*. 2015. URL: https://kaggle.com/competitions/airbnb-recruiting-new-user-bookings.

[34] *Health Insurance Portability and Accountability Act of 1996*. https://www.govinfo.gov/app/details/PLAW-104publ191. Public Law 104–191, 104th Congress (Aug. 21, 1996). 1996.

[35] Dayananda Herurkar, Ahmad Ali, and Andreas Dengel. “Evaluating Generative Models for Tabular Data: Novel Metrics and Benchmarking”. In: *CoRR* abs/2504.20900 (2025). DOI: 10.48550/ARXIV.2504.20900. arXiv: 2504.20900. URL: https://doi.org/10.48550/arXiv.2504.20900.

[36] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

[37] Ari Holtzman et al. “The Curious Case of Neural Text Degeneration”. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=rygGQyrFvH.

[38] Xin Huang et al. “TabTransformer: Tabular Data Modeling Using Contextual Embeddings”. In: *CoRR* abs/2012.06678 (2020). arXiv: 2012.06678. URL: https://arxiv.org/abs/2012.06678.

[39] Valter Hudovernik, Martin Jurkovic, and Erik Strumbelj. “Benchmarking the Fidelity and Utility of Synthetic Relational Data”. In: *CoRR* abs/2410.03411 (2024). DOI: 10.48550/ARXIV.2410.03411. arXiv: 2410.03411. URL: https://doi.org/10.48550/arXiv.2410.03411.

[40] Tero Karras et al. “Elucidating the Design Space of Diffusion-Based Generative Models”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/a98846e9d9cc01cfb87eb694d946ce6b-Abstract-Conference.html.

[41] Jayoung Kim, Chaejeong Lee, and Noseong Park. “STaSy: Score-based Tabular data Synthesis”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/forum?id=1mNssCWt%5C_v.

[42] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014. URL: http://arxiv.org/abs/1312.6114.

[43] Florian Knauer and Will Cukierski. *Rossmann Store Sales*. 2015. URL: https://kaggle.com/competitions/rossmann-store-sales.

[44] Akim Kotelnikov et al. “TabDDPM: Modelling Tabular Data with Diffusion Models”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 17564–17579. URL: https://proceedings.mlr.press/v202/kotelnikov23a.html.

[45] Chaejeong Lee, Jayoung Kim, and Noseong Park. “CoDi: Co-evolving Contrastive Diffusion Models for Mixed-type Tabular Synthesis”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 18940–18956. URL: https://proceedings.mlr.press/v202/lee23i.html.

[46] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *Proceedings of ACL*. 2020, pp. 7871–7880. URL: https://aclanthology.org/2020.acl-main.703/.

[47] Tennison Liu et al. “GOGGLE: Generative Modelling for Tabular Data by Learning Relational Structure”. In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/pdf?id=fPVRcJqspu.

[48] Tongyu Liu et al. “Controllable Tabular Data Synthesis Using Diffusion Models”. In: *Proc. ACM Manag. Data* 2.1 (2024), 28:1–28:29. DOI: 10.1145/3639283. URL: https://doi.org/10.1145/3639283.

[49] Ciro Antonio Mami et al. “Generating Realistic Synthetic Relational Data through Graph Variational Autoencoders”. In: *CoRR* abs/2211.16889 (2022). DOI: 10.48550/ARXIV.2211.16889. arXiv: 2211.16889. URL: https://doi.org/10.48550/arXiv.2211.16889.

[50] Leland McInnes and John Healy. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. In: *CoRR* abs/1802.03426 (2018). arXiv: 1802.03426. URL: http://arxiv.org/abs/1802.03426.

[51] Daniele Micci-Barreca. “A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems”. In: *SIGKDD Explor.* 3.1 (2001), pp. 27–32. DOI: 10.1145/507533.507538. URL: https://doi.org/10.1145/507533.507538.

[52] Jan Motl and Oliver Schulte. *The CTU Prague Relational Learning Repository: FTP Dataset*. https://relational.fel.cvut.cz/dataset/FTP. Accessed: 2025-05-14. 2024. arXiv: 1511.03086 [cs.LG].

[53] R. Kelley Pace and Ronald Barry. “Sparse Spatial Autoregressions”. In: *Statistics and Probability Letters* 33 (1997), pp. 291–297.

[54] Wei Pang et al. “ClavaDDPM: Multi-relational Data Synthesis with Cluster-guided Diffusion Models”. In: *CoRR* abs/2405.17724 (2024). DOI: 10.48550/ARXIV.2405.17724. arXiv: 2405.17724. URL: https://doi.org/10.48550/arXiv.2405.17724.

[55] Noseong Park et al. “Data Synthesis based on Generative Adversarial Networks”. In: *Proc. VLDB Endow.* 11.10 (2018), pp. 1071–1083. DOI: 10.14778/3231751.3231757. URL: http://www.vldb.org/pvldb/vol11/p1071-park.pdf.

[56] Emanuel Parzen. “On estimation of a probability density function and mode”. In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076.

[57] Ekaterina Poslavskaya and Alexey Korolev. “Encoding categorical data: Is there yet anything ’hotter’ than one-hot encoding?” In: *CoRR* abs/2312.16930 (2023). DOI: 10.48550/ARXIV.2312.16930. arXiv: 2312.16930. URL: https://doi.org/10.48550/arXiv.2312.16930.

[58] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://arxiv.org/abs/1511.06434.

[59] Alec Radford et al. *Language Models are Unsupervised Multitask Learners*. Tech. rep. Accessed: 2025-05-21. OpenAI, 2019. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[60] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67.

[61] C. Sakar and Yomi Kastro. *Online Shoppers Purchasing Intention Dataset*. UCI Machine Learning Repository. 2018. DOI: 10.24432/C5F88Q. URL: https://doi.org/10.24432/C5F88Q.

[62] Franco Scarselli et al. “The Graph Neural Network Model”. In: *IEEE Trans. Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605. URL: https://doi.org/10.1109/TNN.2008.2005605.

[63] Jacob Si et al. “TabRep: Training Tabular Diffusion Models with a Simple and Effective Continuous Representation”. In: *arXiv preprint arXiv:2504.04798* (2025).

[64] Jascha Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: *CoRR* abs/1503.03585 (2015). arXiv: 1503.03585. URL: http://arxiv.org/abs/1503.03585.

[65] Aivin V. Solatorio and Olivier Dupriez. “REaLTabFormer: Generating Realistic Relational and Tabular Data using Transformers”. In: *CoRR* abs/2302.02041 (2023). DOI: 10.48550/ARXIV.2302.02041. arXiv: 2302.02041. URL: https://doi.org/10.48550/arXiv.2302.02041.

[66] Gowthami Somepalli et al. “SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training”. In: *CoRR* abs/2106.01342 (2021). arXiv: 2106.01342. URL: https://arxiv.org/abs/2106.01342.

[67] Mihaela Catalina Stoian, Eleonora Giunchiglia, and Thomas Lukasiewicz. “A Survey on Tabular Data Generation: Utility, Alignment, Fidelity, Privacy, and Beyond”. In: *CoRR* abs/2503.05954 (2025). DOI: 10.48550/ARXIV.2503.05954. arXiv: 2503.05954. URL: https://doi.org/10.48550/arXiv.2503.05954.

[68] Hugo Touvron et al. “LLaMA: Open and Efficient Foundation Language Models”. In: *CoRR* abs/2302.13971 (2023). URL: http://arxiv.org/abs/2302.13971.

[69] Allan Tucker et al. “Generating high-fidelity synthetic patient data for assessing machine learning healthcare software”. In: *npj Digit. Medicine* 3 (2020). DOI: 10.1038/S41746-020-00353-9. URL: https://doi.org/10.1038/s41746-020-00353-9.

[70] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[71] Zifeng Wang and Jimeng Sun. “TransTab: Learning Transferable Tabular Transformers Across Tables”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/1377f76686d56439a2bd7a91859972f5-Abstract-Conference.html.

[72] William Wolberg et al. *Breast Cancer Wisconsin (Diagnostic)*. UCI Machine Learning Repository. 1993. DOI: 10.24432/C5DW2B. URL: https://doi.org/10.24432/C5DW2B.

[73] Lei Xu and Kalyan Veeramachaneni. “Synthesizing Tabular Data using Generative Adversarial Networks”. In: *CoRR* abs/1811.11264 (2018). arXiv: 1811.11264. URL: http://arxiv.org/abs/1811.11264.

[74] Lei Xu et al. “Modeling Tabular data using Conditional GAN”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 7333–7343. URL: https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html.

[75] Lei Xu et al. “Modeling Tabular data using Conditional GAN”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 7333–7343. URL: https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html.

[76] Hengrui Zhang et al. “Mixed-Type Tabular Data Synthesis with Score-based Diffusion in Latent Space”. In: *CoRR* abs/2310.09656 (2023). DOI: 10.48550/ARXIV.2310.09656. arXiv: 2310.09656. URL: https://doi.org/10.48550/arXiv.2310.09656.

[77] Benjamin Zi Hao Zhao, Mohamed Ali Kâafar, and Nicolas Kourtellis. “Not one but many Tradeoffs: Privacy Vs. Utility in Differentially Private Machine Learning”. In: *CCSW’20, Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop, Virtual Event, USA, November 9, 2020*. Ed. by Yinqian Zhang and Radu Sion. ACM, 2020, pp. 15–26. DOI: 10.1145/3411495.3421352. URL: https://doi.org/10.1145/3411495.3421352.

[78] Zilong Zhao et al. “CTAB-GAN: Effective Table Data Synthesizing”. In: *Asian Conference on Machine Learning, ACML 2021, 17-19 November 2021, Virtual Event*. Ed. by Vineeth N. Balasubramanian and Ivor W. Tsang. Vol. 157. Proceedings of Machine Learning Research. PMLR, 2021, pp. 97–112. URL: https://proceedings.mlr.press/v157/zhao21a.html.

[79] Zilong Zhao et al. “CTAB-GAN+: Enhancing Tabular Data Synthesis”. In: *CoRR* abs/2204.00401 (2022). DOI: 10.48550/ARXIV.2204.00401. arXiv: 2204.00401. URL: https://doi.org/10.48550/arXiv.2204.00401.

Nomenclature

Abbreviations

Abbreviation	Definition
AUC	Area Under the ROC Curve
ClavaDDPM	Cluster Latent Variable Guided Denoising Diffusion Probabilistic Model
DDPM	Denoising Diffusion Probabilistic Models
DM	Discriminator Measure
DT	Decision Tree
ELBO	Evidence Lower Bound
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GMM	Gaussian Mixture Model
HIPAA	Health Insurance Portability and Accountability Act
KL	Kullback–Leibler
LD	Logistic Detection
LLM	Large Language Model
ML	Machine Learning
MLE	Machine Learning Efficacy
MMD	Maximum Mean Discrepancy
MPNN	Message Passing Neural Network
MSE	Mean Squared Error
MT	Masked Transformer
RDBMS	Relational Database Management System
REaLTabFormer	Realistic Relational and Tabular Transformer
RF	Random Forest
RTF	REaLTabFormer
Seq2Seq	Sequence-to-sequence
VAE	Variational Autoencoder

List of Figures

List of Tables

A

Corrected Tables for Research Paper

In the research paper presented in chapter 2, due to a processing error during evaluation, standard deviation values were reported as 0.00 for most logistic detection results. The corrected values, based on the same underlying data and methods, are presented in this appendix.

Table 2

Table A.1: Relational-dataset evaluation on Discriminator Measure (DM), Logistic Detection (LD) scores, and train duration. Scores are reported as mean$_{\pm \text{standard deviation}}$. For DM, lower is better; for LD, higher is better. Best scores are highlighted in **boldface** and second-best are underlined.

		Metric	Original	Baseline		Flattened	
				RTF	Clava	RTF	Clava
AirBnB	Parent	LD	97.82$_{\pm 1.66}$	52.63$_{\pm 1.55}$	**85.92$_{\pm 1.39}$**	29.24$_{\pm 1.09}$	71.45$_{\pm 1.70}$
		DM	0.02$_{\pm 0.01}$	**0.57$_{\pm 0.05}$**	**0.57$_{\pm 0.05}$**	0.98$_{\pm 0.03}$	1.00$_{\pm 0.00}$
	Child	LD	91.23$_{\pm 1.05}$	42.22$_{\pm 1.65}$	**89.05$_{\pm 0.36}$**	62.30$_{\pm 0.09}$	76.52$_{\pm 0.76}$
		DM	0.05$_{\pm 0.02}$	0.44$_{\pm 0.05}$	0.54$_{\pm 0.07}$	**0.28$_{\pm 0.04}$**	0.88$_{\pm 0.03}$
		T_{train}	-	2762	1049	8891	**407**
Rossman	Parent	LD	95.69$_{\pm 2.08}$	83.58$_{\pm 5.26}$	21.78$_{\pm 1.66}$	**93.77$_{\pm 1.93}$**	82.59$_{\pm 6.64}$
		DM	0.85$_{\pm 0.01}$	0.54$_{\pm 0.02}$	**0.09$_{\pm 0.04}$**	0.49$_{\pm 0.10}$	0.72$_{\pm 0.02}$
	Child	LD	95.19$_{\pm 2.09}$	57.42$_{\pm 0.96}$	88.01$_{\pm 0.68}$	**91.02$_{\pm 0.99}$**	75.51$_{\pm 1.13}$
		DM	0.84$_{\pm 0.05}$	0.40$_{\pm 0.05}$	0.10$_{\pm 0.04}$	**0.09$_{\pm 0.02}$**	0.62$_{\pm 0.06}$
		T_{train}	-	1754	1119	3035	**384**
FTP	Parent	LD	99.12$_{\pm 0.93}$	85.16$_{\pm 1.74}$	**97.54$_{\pm 1.10}$**	OOM	95.74$_{\pm 1.32}$
		DM	0.02$_{\pm 0.01}$	0.11$_{\pm 0.02}$	**0.06$_{\pm 0.04}$**	OOM	0.09$_{\pm 0.04}$
	Child	LD	92.31$_{\pm 0.75}$	29.20$_{\pm 0.83}$	86.55$_{\pm 0.54}$	OOM	**88.02$_{\pm 1.17}$**
		DM	0.49$_{\pm 0.04}$	0.73$_{\pm 0.02}$	0.56$_{\pm 0.02}$	OOM	0.48$_{\pm 0.03}$
		T_{train}	-	1561	961	-	**366**

Table 3

Table A.2: Multi-table relational dataset Logistic Detection (LD) and Discriminator Measure (DM) scores. Relational (Rel.) and Flat correspond to the original relational dataset and the flattened or denormalized (1NF) dataset, respectively. Metric shown as mean$_{\pm \text{standard deviation}}$. For DM, lower is better; for LD, higher is better. **Bold face** indicates best result and underlined second best. We denote training time in seconds with T_{train}.

		Metric	Original	RealTabFormer	Clava	OURS	Ablation* (subsection 5.3)	
							Layer Share	Context ID
AirBnB	Parent *(Users)*	LD	97.82$_{\pm 1.66}$	52.63$_{\pm 1.55}$	85.92$_{\pm 1.39}$	**87.24$_{\pm 2.31}$**	51.55$_{\pm 1.36}$	**87.41$_{\pm 1.84}$**
		DM	0.02$_{\pm 0.01}$	0.57$_{\pm 0.05}$	0.57$_{\pm 0.05}$	0.17$_{\pm 0.06}$	0.57$_{\pm 0.04}$	**0.13$_{\pm 0.02}$**
	Child *(Sessions)*	LD	91.23$_{\pm 1.05}$	42.22$_{\pm 1.65}$	**89.05$_{\pm 0.36}$**	67.18$_{\pm 0.53}$	71.85$_{\pm 0.75}$	32.31$_{\pm 0.69}$
		DM	0.05$_{\pm 0.02}$	0.44$_{\pm 0.05}$	0.54$_{\pm 0.07}$	**0.33$_{\pm 0.07}$**	0.35$_{\pm 0.07}$	0.63$_{\pm 0.05}$
		T_{train}	-	2762	**1049**	3228	4347	4168
Rossmann	Parent *(Stores)*	LD	95.69$_{\pm 2.08}$	83.58$_{\pm 5.26}$	21.78$_{\pm 1.66}$	**86.53$_{\pm 3.60}$**	85.65$_{\pm 8.51}$	**86.53$_{\pm 3.60}$**
		DM	0.85$_{\pm 0.01}$	0.54$_{\pm 0.02}$	**0.09$_{\pm 0.04}$**	0.53$_{\pm 0.01}$	0.54$_{\pm 0.02}$	0.53$_{\pm 0.01}$
	Child *(Sales)*	LD	95.19$_{\pm 2.09}$	57.42$_{\pm 0.96}$	**88.01$_{\pm 0.68}$**	79.30$_{\pm 1.31}$	64.97$_{\pm 0.77}$	77.24$_{\pm 1.88}$
		DM	0.84$_{\pm 0.05}$	0.40$_{\pm 0.05}$	**0.10$_{\pm 0.04}$**	0.27$_{\pm 0.07}$	0.37$_{\pm 0.06}$	0.27$_{\pm 0.07}$
		T_{train}	-	1754	**1119**	3514	1758	3517
FTP	Parent *(Sessions)*	LD	99.12$_{\pm 0.93}$	85.16$_{\pm 1.74}$	**97.54$_{\pm 1.10}$**	85.54$_{\pm 1.12}$	88.44$_{\pm 1.73}$	81.35$_{\pm 0.59}$
		DM	0.02$_{\pm 0.01}$	0.11$_{\pm 0.02}$	**0.06$_{\pm 0.04}$**	0.11$_{\pm 0.02}$	0.10$_{\pm 0.02}$	0.13$_{\pm 0.02}$
	Child *(Products)*	LD	92.31$_{\pm 0.75}$	29.20$_{\pm 0.83}$	**86.55$_{\pm 0.54}$**	71.25$_{\pm 0.70}$	69.19$_{\pm 0.78}$	26.79$_{\pm 0.95}$
		DM	0.49$_{\pm 0.04}$	0.73$_{\pm 0.02}$	0.56$_{\pm 0.02}$	**0.21$_{\pm 0.02}$**	0.62$_{\pm 0.05}$	0.77$_{\pm 0.01}$
		T_{train}	-	1561	**961**	1454	1310	1596

Table 5

Table A.3: Relational-dataset ablation on the impact of contextual information placement on Discriminator Measure (DM), Logistic Detection (LD) scores, and train duration. Idx. refers to the column index position at which the cluster IDs are inserted during ② of our method, where '1' is the default. Scores are reported as mean$_{\pm \text{standard deviation}}$. For DM, lower is better; for LD, higher is better. Best scores shown in **boldface**, second-best are underlined. We denote training time in seconds with T_{train}.

		Metric	Original		Insertion Indx.		
				RTF	Idx. 0	Idx. 1	Last Idx.
AirBnB	Parent	LD	97.82$_{\pm 1.66}$	52.63$_{\pm 1.55}$	29.24$_{\pm 0.00}$	**78.99$_{\pm 1.05}$**	47.56$_{\pm 1.01}$
		DM	0.02$_{\pm 0.01}$	0.57$_{\pm 0.05}$	0.98$_{\pm 0.03}$	**0.22$_{\pm 0.04}$**	0.62$_{\pm 0.02}$
	Child	LD	91.23$_{\pm 1.05}$	42.22$_{\pm 1.65}$	**62.30$_{\pm 0.00}$**	50.64$_{\pm 0.77}$	59.06$_{\pm 0.69}$
		DM	0.05$_{\pm 0.02}$	0.44$_{\pm 0.05}$	**0.28$_{\pm 0.04}$**	0.42$_{\pm 0.05}$	0.39$_{\pm 0.05}$
		T_{train}	-	2762	2823	**2004**	2575
FTP	Parent	LD	99.12$_{\pm 0.93}$	85.16$_{\pm 1.74}$	84.63$_{\pm 1.55}$	85.54$_{\pm 1.12}$	**85.60$_{\pm 1.01}$**
		DM	0.02$_{\pm 0.01}$	**0.11$_{\pm 0.02}$**	0.13$_{\pm 0.03}$	**0.11$_{\pm 0.02}$**	**0.11$_{\pm 0.02}$**
	Child	LD	92.31$_{\pm 0.75}$	29.20$_{\pm 0.83}$	57.96$_{\pm 0.63}$	**71.25$_{\pm 0.70}$**	**68.02$_{\pm 0.77}$**
		DM	0.49$_{\pm 0.04}$	0.73$_{\pm 0.02}$	0.60$_{\pm 0.03}$	**0.21$_{\pm 0.02}$**	0.26$_{\pm 0.02}$
		T_{train}	-	1561	1707	**1454**	1584

Table 6

Table A.4: Relational-dataset evaluation on AirBnB for Discriminator Measure (DM), Logistic Detection (LD) scores, and train duration T_{train}. 'Learned Spaces' indicate the embedding space used to assign cluster ID in step ① of our method. Scores are reported as mean$_{\pm\text{standard deviation}}$. For DM, lower is better; for LD, higher is better. Best scores are highlighted in **boldface** and second-best are underlined.

		Metric	Original		Learned Spaces	
				RTF	TransTab	One-Hot
AirBnB	Parent	LD	97.82$_{\pm 1.66}$	52.63$_{\pm 1.55}$	78.07$_{\pm 2.22}$	**87.24**$_{\pm \mathbf{2.31}}$
		DM	0.02$_{\pm 0.01}$	0.57$_{\pm 0.05}$	0.21$_{\pm 0.03}$	**0.17**$_{\pm \mathbf{0.06}}$
	Child	LD	91.23$_{\pm 1.05}$	42.22$_{\pm 1.65}$	56.31$_{\pm 0.63}$	**67.18**$_{\pm \mathbf{0.53}}$
		DM	0.05$_{\pm 0.02}$	0.44$_{\pm 0.05}$	0.35$_{\pm 0.05}$	**0.33**$_{\pm \mathbf{0.07}}$
		T_{emb}	-	-	2735	1
		T_{train}	-	2762	3237	3228
		T_{total}	-	**2762**	5972	3229