On the Gap Between Diffusion and Transformer Multi-Tabular Generation

Paardekooper, Gijs; Galjaard, Jeroen M.; Chen, Lydia Y.

DOI
10.1145/3746252.3761530

Publication date
2025

Document Version
Final published version

Published in
CIKM 2025 - Proceedings of the 34th ACM International Conference on Information and Knowledge Management

On the Gap Between Diffusion and Transformer Multi-Tabular Generation

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

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

Lydia Y. Chen
lydiaychen@ieee.org
University of Neuchâtel
Neuchâtel, Switzerland

Abstract

Shareable tabular data is of high importance in industry and research. While generating synthetic records is well-studied, research has only recently extended to *relational data synthesis*. In the tabular generation setting, diffusion and transformer models exhibit superior performance over prior art. However, in the relational setting, diffusion models outperform transformers. This work focuses on the performance gap between tabular transformers and diffusion models in 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 single- and multi-table datasets, highlighting the gap's root causes between the methods. In our experiments, we attribute this difference to the influence of contextual information and data representation. To bridge the gap in the relational setting, we propose two seemingly simple strategies: layer sharing and contextual cues. This work[1] offers insights into key design considerations for single- and multi-table generative models, including the incorporation of contextual information and the reuse of existing knowledge. With the proposed methods, we achieve improvements of 1.52× and 1.94× for the Logistic Detection and Discriminator Measure metrics, respectively.

CCS Concepts

• **Information systems** → *Data management systems*; Data mining; *Clustering*; • **Theory of computation** → **Data modeling**; • **Computing methodologies** → **Structured outputs**; **Neural networks**.

Keywords

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

ACM Reference Format:
Gijs Paardekooper, Jeroen M. Galjaard, and Lydia Y. Chen. 2025. On the Gap Between Diffusion and Transformer Multi-Tabular Generation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3746252.3761530

[1]Code available online https://gitlab.ewi.tudelft.nl/dmls/publications/minding-the-gap.

CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761530

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 in the image domain, Variational Auto-encoder (VAE) and GAN-based approaches, such as TVAE and CTGAN [31], have emerged. 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

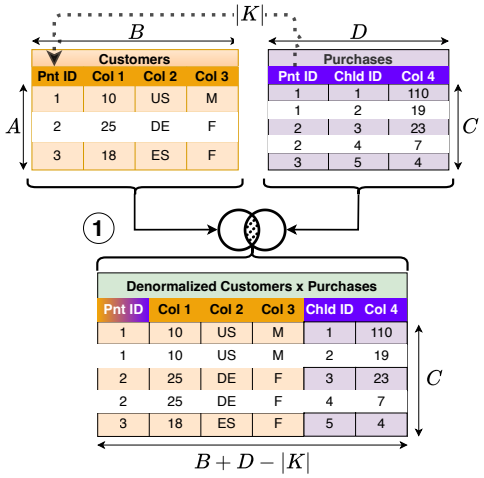

Figure 1: Example relational dataset with 'customers' as Parent (Pnt) table and 'purchases' as subsequent Child (Chld) table. Step ① illustrates creating a denormalized dataset by performing an inner-join on the primary key K 'Pnt ID', to flatten relational data. The pairs A, C and C, D, represent the 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.

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

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. 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 refer to as '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.

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.

Table 2: Relational-dataset evaluation on Discriminator Measure (DM), Logistic Detection (LD) scores, and training duration (T_{train}). 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. Out of Memory (OOM) indicates an inability to complete.

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

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. Subsequentially, we investigate the impact of *representing the relational information*, by denormalizing the relational dataset as a single (flattened) table. Therein, 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, 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. Our proposed change is two-fold. Firstly, by changing the relational data representation to the base adapter. Secondly, providing hints to the base and adapter models to simplify the conditional distribution of relational records. Towards the latter, we construct 'in-context' hints for the transformer model during 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. Firstly, in step ① we cluster on the joint parent (X) and child (Y) tables as shown in Algorithm 1. Secondly, in step ② the preceding information is inserted as a categorical column in X and Y, at column index q. Afterwards, we 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 non-embedding parameters. As both models now inherit their parameters from the Base model, only the Adapter model's Child vocabulary can be learned on the Child's data.

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 modeling. Given that classifier-based diffusion on its own is not directly portable to an auto-regressive architecture, we turn to in-context representation. Similar to classifier-based guidance, this makes the conditional 'guidance' token learnable before it is used during inference. Therewith, providing a signal to the generator to conditionally sample from a simplified distribution. Hereby, allowing the generator to learn to steer the generation process based on the cluster identifier (ID) token. Moreover, enabling conditional sampling from the cluster ID by providing the model with the cluster ID as context during inference. As a result, providing *classifier-based guidance* to the auto-regressive generative model.

To create the contextual information for 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 a configurable

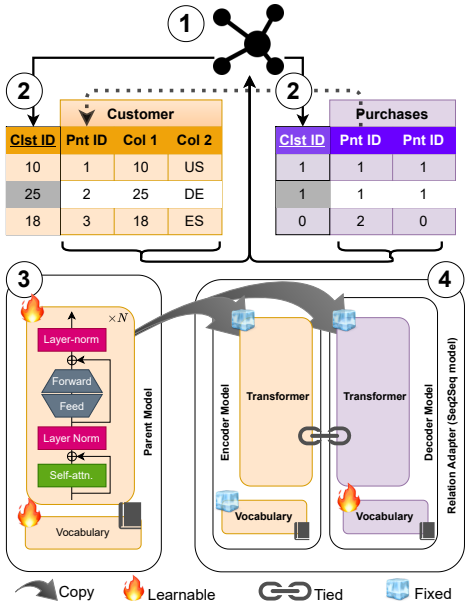

Figure 2: Representing tabular data with contextual information, orange and purple indicate Parent and Child data/model respectively. Contextual information is added to related Parent and Child records using clustering in steps ①–②. Subsequently, a Base model is trained on the Parent Table (③). Following,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

Algorithm 1 Cluster-Augmented Model Training corresponding to steps ① – ④ 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.

Step ①: Cluster encoded data

$\mathbf{X}, \mathbf{Y} \leftarrow D_{\text{train}}$ {Split Parent and Child.}

$\mathbf{C}_X, \mathbf{C}_Y \leftarrow \texttt{Cluster}(\mathbf{X}, \mathbf{Y}, k)$ {Get Parent & Child assignment.}

Step ②: Add contextual information

{Insert context for rows $\mathbf{X}, \mathbf{Y}$ with cluster ID (①) at position q}

$X \leftarrow \left[\texttt{insert}(\mathbf{C}_X[o], \mathbf{X}[o], q) \mid o \in [1, \ldots, |\mathbf{X}|]\right]$

$Y \leftarrow \left[\texttt{insert}(\mathbf{C}_Y[j], \mathbf{Y}[j], q) \mid j \in [1, \ldots, |\mathbf{Y}|]\right]$

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 {Tie Transformer block weights}

$\mathbf{X} \leftarrow$ Base($\mathbf{X}$)

$\mathbf{Y} \leftarrow \texttt{Tokenize}(\mathbf{Y})$

Adapter $\leftarrow$ Train(Adapter, $\mathbf{X}, \mathbf{Y}$) {Finetune the vocabulary.}

Returns: Base, Adapter

insertion position, 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 IDs between methods. In our experiments, we run the clustering algorithm on standardized numerical and one-hot encoded categorical features, enabling clustering in the 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 subsequently learned separately to predict a child's '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 with a random initialization. Subsequently, we tokenize the Parent and Child dataset on which the models are trained. First, we train the Base model on the augmented Parent, in step ③. Afterwards, we continue by fine-tuning the Adapter module.

Before fine-tuning the relational Adapter, we reduce its number of learnable parameters compared to REaLTabFormer, as shown ④ in Figure 2. Here, we perform layer-wise weight sharing of the Transformer block weights between the Base and Adapter models. 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 optimize only the parameters corresponding to the tokens that represent the Child rows. Thus, refraining from further tuning the tied Parent and Child module's non-embedding parameters. As a result, we are left with $\sim$ 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. Herein, we remark that layer sharing in the NLP domain has been applied successfully [30, 33–35].

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 presenting the experimental results, let us outline the standard 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 has only a minimal impact on 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.

5.3 Ablation: Evaluating Layer Sharing and Contextual Cues

Next, we consider the impact of the two design considerations employed in our method independently. We summarize these results

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}. We denote the time required to train the models by T_{train}. Flattened column denoted with † represent reprinted results from Table 2 for the reader's convenience.

		Metric	Original	RealTabFormer	Clava	OURS	Ablation* (subsection 5.3)		
							Layer Share	Context ID	Flattened†
AirBnB [10]	Parent	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**	29.24±0.00
	(Users)	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**	0.98±0.03
	Child	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	62.30±0.00
	(Sessions)	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	0.28±0.04
		T_{train}	-	2762	1049	3228	4347	4168	8891
Rossmann [14]	Parent	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**	94.42±0.00
	(Stores)	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	0.49±0.01
	Child	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	90.84±0.00
	(Sales)	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	0.09±0.00
		T_{train}	-	1754	1119	3514	1758	3517	3035
FTP [18]	Parent	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	OOM
	(Sessions)	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	OOM
	Child	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	OOM
	(Products)	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	OOM
		T_{train}	-	1561	961	1454	1310	1596	-

in the 'Ablation' column of Table 3, where we also reprint the 'Flattened' results from Table 2. We focus on the contribution of layer sharing (Layer Share) and the contextual cue (Context ID) independently of the relational datasets.

Herein, we observe similar trends to the 'implicit' layer sharing found in the data-flattening experiments. Through explicit 'Layer Share'-ing, the Parent table's performance relative to Flattened is close to (Rossman) or even exceeds (AirBnB) it. Moreover, on the Child performance we see that explicit layer sharing improves considerably over RTF, while the Flattened representation can still be beneficial. Augmenting the data with a context ID boosts RTF—except on FTP's 'products' table. Similarly, the *joint application* of contextual information and layer sharing consistently improves the generation of transformer-based relational data. However, we see that the joint application of layer sharing and contextual information outperforms the independent application thereof.

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 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

Table 5: Relational-dataset ablation on insertion index (Idx.) of contextual information in ② on Discriminator Measure (DM), Logistic Detection (LD) scores, and train duration. Here, insertion at index '1' is the default. Scores are reported as mean$_{\pm \text{standard deviation}}$. For DM, lower is better; for LD, higher is better. Training time is denoted in seconds as T_{train}. Best scores shown in boldface, second-best are underlined.

		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

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 Ablating Clustering Representation

Lastly, we consider the impact of altering the data representation on which the clustering algorithm is run. Rather than directly clustering on the data domain, we first create a learned tabular embedding space. Using an external embedding space enables the unsupervised enrichment of tabular data by capturing complex, high-order relationships that facilitate clustering methods. While such models enable the creation of high-fidelity embeddings of heterogeneous data [12, 25, 28], 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.

In Table 6, we compare the performance differences between the various clustering methods on the Airbnb dataset. While both embeddings improve over REaLTabFormer performance, the One-Hot provides superior performance. Moreover, it requires considerably less compute, adding around a single second to the training process. Although additional embedding mechanisms could be considered, given the already strong performance of the baseline method, we leave this for future work.

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

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.

Acknowledgments

This research is part of the Priv-GSyn project, 200021E_229204 of Swiss National Science Foundation and the DEPMAT project, P20-22 / N21022, of the research programme Perspectief, which is partly financed by the Dutch Research Council (NWO).

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* (2024). arXiv:2401.02524 doi:10.48550/ARXIV.2401.02524

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

[3] R. Bock. 2004. MAGIC Gamma Telescope. UCI Machine Learning Repository. 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 Int'l Conf. 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.* 1 (2001), 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.

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (San Francisco, CA, USA) *(KDD '16)*. ACM, NY, USA, 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.* 1 (1997), 119–139. doi:10.1006/JCSS.1997.1504

[9] Jerome H. Friedman. 2001. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics* 5 (2001), 1189–1232. http://www.jstor.org/stable/2699986

[10] Alok Gupta, Anna Montoya, Liz Sellier, Meghan O'Connell, and Wendy Kan. 2015. Airbnb New User Bookings. 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 Conf. 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.). Curran Associates, Inc., 6840–6851. 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* (2020). arXiv:2012.06678 https://arxiv.org/abs/2012.06678

[13] Valter Hudovernik, Martin Jurkovic, and Erik Strumbelj. 2024. Benchmarking the Fidelity and Utility of Synthetic Relational Data. *CoRR* (2024). arXiv:2410.03411 doi:10.48550/ARXIV.2410.03411

[14] Florian Knauer and Will Cukierski. 2015. Rossmann Store Sales. 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 *Int'l Conf. on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, HI, USA (Proceedings of Machine Learning Research)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). 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 *Int'l Conf. on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, MD, USA (Proceedings of Machine Learning Research)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). 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 Int'l Conf. 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. arXiv:1511.03086 [cs.LG] Accessed: 2025-05-14.

[19] R. Kelley Pace and Ronald Barry. 1997. Sparse Spatial Autoregressions. *Statistics and Probability Letters* (1997), 291–297.

[20] Wei Pang, Masoumeh Shafieinejad, Lucy Liu, and Xi He. 2024. ClavaDDPM: Multi-relational Data Synthesis with Cluster-guided Diffusion Models. *CoRR* (2024). 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. 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* 1 (2009), 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. arXiv:2504.16506 [cs.LG] https://arxiv.org/abs/2504.16506

[24] Aivin V. Solatorio and Olivier Dupriez. 2023. REaLTabFormer: Generating Realistic Relational and Tabular Data using Transformers. *CoRR* (2023). 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* (2021). arXiv:2106.01342 https://arxiv.org/abs/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* (2025). 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 Conf. 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.). Curran Associates, Inc., 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 Conf. 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.). Curran Associates, Inc., 2902–2915. http://papers.nips.cc/paper_files/paper/2022/hash/1377f76686d56439a2bd7a91859972f5-Abstract-Conf..html

[29] William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. 1993. Breast Cancer WI (Diagnostic). UCI Machine Learning Repository. 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 Conf. on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conf., IAAI 2019, The Ninth AAAI Sym. on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, HI, 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 Conf. 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.). Curran Associates, Inc., 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 Conf. 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.). Curran Associates, Inc., 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 (Vol. 1: Long Papers)*, Iryna Gurevych and Yusuke Miyao (Eds.). Association for Computational Linguistics, Melbourne, Australia, 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* (2016).

[35] Long Zhou, Yuchen Liu, Jiajun Zhang, Chengqing Zong, and Guoping Huang. 2018. Language-Independent Representor for Neural Machine Translation. *CoRR* (2018). arXiv:1811.00258 http://arxiv.org/abs/1811.00258