AutoDiff: combining Auto-encoder and Diffusion model for tabular data synthesizing

Namjoon Suh
UCLA
namjsuh@ucla.edu

Xioafeng Lin
UCLA
bernardo1998@g.ucla.edu

Din-Yin Hsieh
UCLA
darrenhsieh1205@g.ucla.edu

Merhdad Honarkhah
Meta
merhdad@meta.com

Guang Cheng
UCLA
guangcheng@stat.ucla.edu

Abstract

Diffusion model has become a main paradigm for synthetic data generation in many subfields of modern machine learning, including computer vision, language model, or speech synthesis. In this paper, we leverage the power of diffusion model for generating synthetic tabular data. The heterogeneous features in tabular data have been main obstacles in tabular data synthesis, and we tackle this problem by employing the auto-encoder architecture. When compared with the state-of-the-art tabular synthesizers, the resulting synthetic tables from our model show nice statistical fidelities to the real data, and perform well in downstream tasks for machine learning utilities. We conducted the experiments over 15 publicly available datasets. Notably, our model adeptly captures the correlations among features, which has been a long-standing challenge in tabular data synthesis. Our code is available at https://github.com/UCLA-Trustworthy-AI-Lab/AutoDiffusion.

1 Introduction

The creation of synthetic tabular data is invaluable for research, testing, and analysis, especially when real-world data is scarce or sensitive. It facilitates scenario exploration, algorithm testing, and practical data analysis experiences for students and professionals. Additionally, synthetic tabular data serves as a benchmark for evaluating data processing and predictive models, ensuring safe performance assessment. It addresses privacy, data scarcity, and accessibility issues, offering new possibilities for data-driven research in academia and industry.

Given the importance of synthesizing tabular data, many researchers have put enormous efforts on building tabular synthesizers with fidelity and utility guarantees. CTGAN [24] and its variants [25, 26] (e.g., CTABGAN, CTABGAN+) have gained popularity for generating tabular data using a Generative Adversarial Networks [8] (GANs). These models employ advanced data encoders, modeling continuous variables with Gaussian Mixture models (GMM) and discrete variables with one-hot encoding. However, GMM may not work well for certain real-world continuous variables, and one-hot encoding can increase data dimensionality for discrete variables, requiring large neural networks.

With the rise of Dalle-2 [20], diffusion models [23] have excelled, outperforming Generative Adversarial Network (GAN)[8] models in various domains like image synthesis[4], medical imaging [17], etc. Recently, diffusion-based tabular synthesizers, like Stasy [11], have shown promise by preprocessing data with min-max scaling and one-hot encoding, outperforming GAN-based methods in various tasks. Yet, score-based diffusion models [23] weren't initially designed for heterogeneous features.

NeurIPS 2023 SyntheticData4ML workshop.

Newer approaches, such as those using Doob's h-transform [15], TabDDPM [13], and CoDi [14], aim to address this challenge by combining different diffusion models [23, 10] or leveraging contrastive learning [22] to co-evolve models for improved performance on heterogeneous data. Following this line of research, we present a new tabular data synthesizer combining the ideas of auto-encoder and diffusion model referred as **AutoDiff**. In the next subsection, we specify the contributions of our model in the context of challenges of tabular synthesizing.

1.1 Main Contributions

In this subsection, we present three challenges of tabular data synthesizing and the main ideas of our model **AutoDiff** to tackle those issues. To facilitate the understandings of AutoDiff model, the ideas are often highlighted by brief comparisons with the other state-of-the-art (SOTA) models

Heterogeneous features are the most challenging issue for building a tabular synthesizer, as real-world tables frequently have numerical, discrete, or even mixed-type features. We are tackling this challenge by combining the ideas from auto-encoder and score-based diffusion model [23]. An autoencoder is a type of neural network architecture that learns to encode input data into latent representations and reconstruct them back to the original input. We leverage the power of autoencoder for learning the *"continuous respresentations"* of the original heterogeneous features in the latent space. Then, the learned representations are fed to diffusion model to generate new latent representations. The trained decoder translates the newly generated representations back to the form of original heterogeneous features. This idea nicely takes the advantages of both models as auto-encoder can deal with the heterogeneous features and diffusion model has been showing the great performances for learning the distributions on continuous space.

Mixed-type feature is a type of features commonly observed in many real-world tabular data, where it has both numerical and discrete components. These features are specifically difficult to capture as most of generative models only focus on either learning distributions on numerical or discrete domains. So it is not clear by simply combining the two different models which are separately designed for numerical and discrete variables. In this paper, we tackle this issue by creating a dummy variable which encodes the frequency of repeated values in a mixed-type feature. This dummy variable is appended to the pre-processed table as an input to the autoencoder so that the latent representations of autoencoder include the information about the dummy variable. The new latent representations generated from diffusion model are decoded back through pre-trained decoder. The output of decoder has both mixed-type feature and corresponding dummy feature. The information of these two features are combined together as the final output. More detailed descriptions are given in section 2. It should be noted the mixed-type encoder introduced in [25] is different from ours in a sense that they directly encode the rows via one-hot-encoding and parameters in GMM. To showcase the performance of our idea, we present comparison plots of generated versus real data of a mixed-type variable over 6 different models in Figure 1.

Correlations of features are important statistical objects that need to be captured when building a tabular synthesizer. Yet, due to the heterogeneous nature of tabular data, capturing these correlations is even more demanding than capturing correlations among purely numerical or discrete features. AutoDiff naturally circumvents this challenge by its construction as diffusion model learns the joint distribution of latent representations in the *continuous* space. As long as auto-encoder in AutoDiff gives good latent representations of rows in the table, it should be expected AutoDiff can nicely capture the correlations among features. We would like to emphasize this contribution by contrasting our idea with the SOTA diffusion based method, TabDDPM, where each categorical variables are considered as independent (i.e., they use separate forward diffusion process for each categorical variable.) and the categorical and numerical variables are also modelled as independent since they used two different types of diffusion models for numerical and discrete variables, respectively. Similar arguments can be made for GAN-based methods (i.e., CTGAN, CTABGAN+) as they separately employ the GMM for numerical features. We perform comparative experiments of these models with ours on various real world datasets whose results are given in Table 1 in section 3.

Numerical comparisons of AutoDiff with other models (with publicly available codes), namely, CTGAN [24], TVAE [24], CTABGAN+ [26], Stasy [11], and TabDDPM [13] are conducted comprehensively across 15 real-world datasets under various metrics. In addition to the models mentioned above, we introduce the AutoGAN model, which is a customized adaptation of MedGAN [3]. This inclusion is specifically tailored for the purpose of comparing diffusion model's performance with

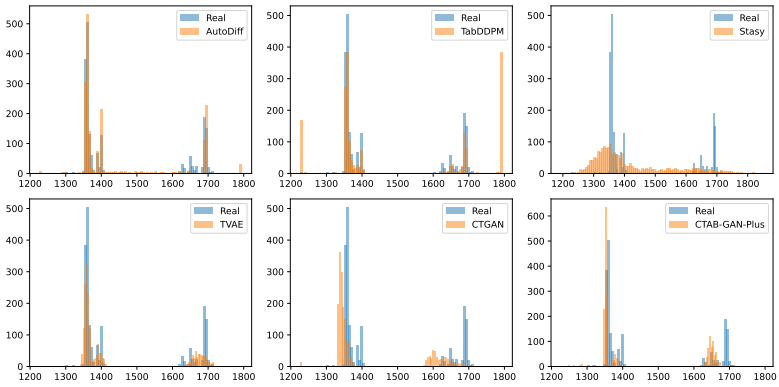

Figure 1: Comparison histograms of a generated and real "Mixed-type" feature over 6 models. The feature presented in the plot is *"Length of Conveyer"* in faults dataset.

GAN model's under same auto-encoder. Specifically, qualities of generated tables are measured through (1) statistical fidelities, (2) machine learning utilities in downstream tasks, and (3) privacy guarantees via Distances to Closest Records (DCR) in [25].

2 Proposed Method

In this section, we provide detailed descriptions on each component in our model. Pre-and post-processing steps of the input and synthesized tabular data are introduced. Then, constructions on auto-encoder and diffusion models are provided.

Pre- and post-processing steps. It is essential to pre-process the real tabular data in a form that the machine learning model can extract the desired information from the data properly. We divide the heterogeneous features into three categories; (1) numerical, (2) discrete, and (3) mixed-type features. Following is how we categorize the variables, and process each feature type. Let $\mathbf{x}$ be the column of a table to be processed.

1. ***Numerical feature*** : If $\mathbf{x}$'s entries are real-valued continuous, we categorize $\mathbf{x}$ as a numerical feature. Moreover, if the entries are integers with more than 25 distinct values, e.g., "Age in adult-income dataset", the $\mathbf{x}$ is categorized as a numerical variable. Here, 25 is a user-specified threshold. We will be using either min-max scaler or gaussian quantile transformation from scikit-learn library [19] for pre-processing numerical features. Hereafter, we denote $\mathbf{x}_{\text{Num}}^{\text{Proc}}$ as the processed column.
2. ***Discrete feature*** : If $\mathbf{x}$'s entries have string datatype, we categorize $\mathbf{x}$ as a discrete feature, e.g., "Gender". Additionally, the $\mathbf{x}$ with less than 25 distinct integers is categorized as a discrete feature. For pre-processing, we simply map the entries of $\mathbf{x}$ to the integers greater than or equal to 0, and further divide the data type into two parts; binary and categorical, denoting them as $\mathbf{x}_{\text{Bin}}^{\text{Proc}}$ and $\mathbf{x}_{\text{Cat}}^{\text{Proc}}$.
3. ***Mixed-type feature*** : Let's say $\mathbf{x}$ is categorized as a numerical feature. If certain values in the $\mathbf{x}$ are repeated more than h-percentages of the entire data points, we consider the $\mathbf{x}$ as the mixed-type feature. Here, h is a user-specified parameter, and we set it as 1 throughout this paper. The idea to encode the mixed-type variable is to create another discrete feature variable $\mathbf{y}$, which encodes the labels of entries in $\mathbf{x}$. We label the entries repeated less than h-percent as 0s. For entries repeated more than h-percent, we label them with the integers from 1 to K in the order of frequent repetition. Here, K is the total number of entries which are repeated more than h-percent of the entire data points.
4. ***Post-processing step*** : After the AutoDiff model generates a synthetic dataset, it must be restored to its original format. For numerical features, this is achieved through inverse transformations, such as reversing min-max scaling or using inverse normal transformations. Integer labels in discrete features are mapped back to their original categorical or string values. For mixed-type features, denote $\mathbf{y}^{\text{syn}}$ as the newly synthesized column from the Au-

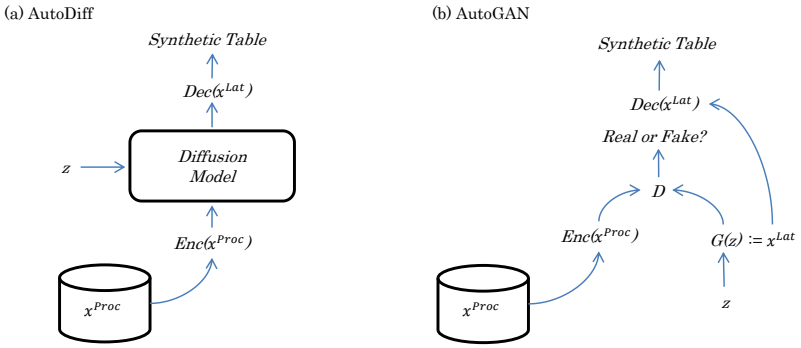

Figure 2: Structures of AutoDiff and AutoGAN models. Here z are random inputs to the models for newly generated latent vectors $\mathbf{x}^{\text{Lat}}$.

toDiff model of the dummy variable $\mathbf{y}$ generated in the item 3. We combine the information of $\mathbf{x}^{\text{syn}}$ and $\mathbf{y}^{\text{syn}}$ leaving the entries in $\mathbf{x}^{\text{syn}}$ as they are, if the corresponding labels in $\mathbf{y}^{\text{syn}}$ are 0. If the labels in $\mathbf{y}^{\text{syn}}$ are the integer values greater than 0, then the corresponding entries in $\mathbf{x}^{\text{syn}}$ are replaced with the repeated values used to encode $\mathbf{y}$.

Auto-encoder. The pre-processed input data $\mathbf{x}^{\text{Proc}} := [\mathbf{x}^{\text{Proc}}_{\text{Num}}; \mathbf{x}^{\text{Proc}}_{\text{Bin}}; \mathbf{x}^{\text{Proc}}_{\text{Cat}}]$ is fed into the auto-encoder i.e., AE, and AE learns the latent representations of the input data. Network architecture of the AE simply consists of multilayer perceptron blocks with ReLU activation function.

$$\text{MLPBlock}(\mathbf{x}) = \text{ReLU}(\text{Linear}(\mathbf{x}))$$

$$\mathbf{x}^{\text{Lat}} = \text{Linear}(\text{MLPBlock}(\cdots \text{MLPBlock}(\mathbf{x}^{\text{Proc}}))) \tag{1}$$

$$\mathbf{x}^{\text{Out}} = \text{Linear}(\text{MLPBlock}(\cdots \text{MLPBlock}(\mathbf{x}^{\text{Lat}}))) \tag{2}$$

Let $\mathbf{x}^{\text{Out}}_{\text{Num}}$, $\mathbf{x}^{\text{Out}}_{\text{Bin}}$, and $\mathbf{x}^{\text{Out}}_{\text{Cat}}$ be the numerical, binary, and categorical outputs of the decoder (2). Note that the dimensions of $\mathbf{x}^{\text{Out}}_{\text{Num}}$ and $\mathbf{x}^{\text{Out}}_{\text{Bin}}$ are same with their corresponding inputs, but the dimension of $\mathbf{x}^{\text{Out}}_{\text{Cat}}$ is $\sum_i K_i$ where K_i is the number of categories in each categorical variable. We set the dimensions of output features in this way as we used the mean-squared (MSE), binary cross entropy (BCE), and cross entropy (CE) in Pytorch package. Additionally, in order to keep the domain of the original numerical features, (e.g., "Age" cannot be negative.) we truncate the output of $\mathbf{x}^{\text{Out}}_{\text{Num}}$ with min and max values of $\mathbf{x}^{\text{Proc}}_{\text{Num}}$. Finally, we minimize the following objective function:

$$\text{MSE}(\mathbf{x}^{\text{Proc}}_{\text{Num}}, \mathbf{x}^{\text{Out}}_{\text{Num}}) + \text{BCE}(\mathbf{x}^{\text{Proc}}_{\text{Bin}}, \mathbf{x}^{\text{Out}}_{\text{Bin}}) + \text{CE}(\mathbf{x}^{\text{Proc}}_{\text{Cat}}, \mathbf{x}^{\text{Out}}_{\text{Cat}}). \tag{3}$$

It should be noted that the dimension of $\mathbf{x}^{\text{Lat}}$ is kept same with that of $\mathbf{x}^{\text{Proc}}$ and the outputs of all the other layers are set as 250 throughout the paper.

Stasy- & Tab-AutoDiff. The score-based diffusion model [23] is utilized to generate new latent representations of $\mathbf{x}^{\text{Lat}}$. Due to space limitations, we defer the detailed descriptions of the diffusion model to Appendix B. In this work, we utilize Variance Preserving (VP)-SDE for data perturbation, and Euler-Maruyama method [23] for sampling over the experiments. Here, we devise two types of AutoDiff models. **Stasy-AutoDiff** (resp. **Tab-AutoDiff**) adopts the same time-dependent score network used in Stasy [11] (resp. TabDDPM [13]) and min-max scaler (resp. Gaussian Quantile Transformation) is used for pre-processing numerical variables. The constructions of the score networks with detailed architectural settings for two models are deferred to Appendix B. The reason for employing the same score networks is to facilitate comparisons of the performances between the Stasy-AutoDiff (resp. Tab-AutoDiff) and Stasy (resp. TabDDPM). (The näive form of Stasy without fine-tuning step is employed in this work.) The overall pipeline of AutoDiff is given in Figure 2.

Med-AutoDiff & AutoGAN. MedGAN [3] has a very similar structure with AutoDiff in a sense that they used the idea of combining auto-encoder and GAN model. The model is only applicable to datasets with discrete variables. Notably, they used MSE for categorical variables instead of CE-loss in (3). Following this set-up, we modify the loss (3) and refer the trained model as Med-AutoDiff. Furthermore, we design AutoGAN model which is applicable to heterogeneous features. We take the advantage of auto-encoder mentioned above, and used the same generator and discriminator in GAN

Models	Marginal		Joint		
	Num Dist. (Wasserstein)	Cate Dist. (Jensen Shannon)	Num-Num (Pearson Corr)	Cat-Cat (Theil's U)	Num-Cat (Correl Ratio)
CTGAN	5.10	**3.86**	1.60	0.27	0.35
TVAE	4.05	5.77	0.93	0.70	0.30
CTABGAN+	4.63	4.61	1.59	0.39	0.53
Stasy	7.95	5.23	2.88	0.26	0.56
TabDDPM	4.42	5.11	2.16	0.35	0.60
AutoGAN	6.49	5.23	0.78	1.65	0.62
Stasy-AutoDiff	**3.57**	4.42	0.66	**0.15**	**0.15**
Med-AutoDiff	3.67	5.32	**0.46**	0.69	0.37
Tab-AutoDiff	3.92	5.32	0.81	**0.15**	0.17

Table 1: Columns $1, 2$: Fidelity ranks of marginal distributions over all datasets in terms of Wasserstein distance between numerical columns and Jensen-Shannon divergence between categorical columns. Columns $3, 4, 5$: L^2-differences between correlation matrices computed on real and synthesized tables under three metrics.

model introduced in MedGAN [3]. Specifically, we use the minibatch averaging for discriminator and batch normalization and skip connection for generator. Same network architectures are used for the constructions of networks as in [3]. The most notable difference between AutoGAN and MedGAN is from the arguments taken in discriminator: in AutoGAN, discriminator distinguishes the real from fake **latent** vectors, whereas in MedGAN, it distinguishes the real from fake **decoded** latent vectors, and finetune the decoder while training discriminator. We choose to fix the parameters in decoder for the fair comparisons with Med-AutoDiff. See Figure 2. Also note that we use the same auto-encoder loss of Med-AutoDiff in AutoGAN.

3 Numerical Experiments & Future Works

Datasets. We collect 15 different real datasets to run the quantitative comparison experiments of AutoDiff model. These datasets are commonly employed to test the performances of recent tabular data synthesizers. Links for the datasets are provided in Appendix A. Across all 6 models, 10 synthetic tables for each of the 15 datasets are generated.

Fidelity Comparisons. Fidelity of the synthesized table refers to how well the generated table retains the statistical properties of the original dataset. Inspired from [25, 13], we assess the similarities in marginal and joint feature distributions between the real (**R**) and synthetic (**S**) tables using a range of metrics. Specifically, for numerical features, we employ the Wasserstein distance (**WD**) to quantify the differences between **R** and **S**. For categorical features, we use Jensen-Shannon (**JS**) divergence to gauge the similarity in their distributions. The first two columns in Table 1 record the averaged rankings of models for the respective dissimilarity measures applied to the columns between the **R** and **S** across all 150 datasets considered for evaluation. Additionally, we evaluate joint distribution quality using various methods depending on the feature types. This includes using the Pearson correlation coefficient for numerical-numerical feature relationships, Theil's U statistics for assessing the dependence between categorical-categorical features, and the correlation ratio for categorical-numerical feature associations. The last three columns in Table 1 record the averaged L^2-distance of computed correlation matrices between $\mathbf{R}$ and $\mathbf{S}$ across 150 synthesized tables for each model. The result shows that the three AutoDiff models show either the best results (bold) or the second best results (underlined) for all 5 metrics. It turns out that the AutoDiff recovers the numerical features pretty well, but performs relatively poor on the recovery of categorical features, when compared with CTGAN. The good performances on **WD** metric is due from the combination of sophisticated design of auto-encoder, specifically for the mixed-type features, and the power of diffusion model for its ability to learn continuous variables. The role of the auto-encoder is even manifested when the result is compared with Stasy and TabDDPM. Recall that exactly the same diffusion models with Stasy-AutoDiff and Tab-AutoDiff are employed in Stasy and TabDDPM. Turning our attentions to correlation measures, three AutoDiff models shows the best performances for the three metrics as presented in the Table 1. This matches with our expectations as diffusion model in AutoDiff is designed to capture the joint distribution of $\mathbf{x}^{\text{Lat}}$ unlike other models where they employ different models for different data types. Stasy turns out to perform well at capturing the correlations among categorical variables, and TVAE is good at capturing Num-Cat correlations.

Models	Classification			Regression	
	Accuracy	**F1**	**AUROC**	R^2	**RMSE**
Identity	0.923	0.809	0.941	0.633	1817.307
CTGAN	0.699	0.465	0.742	0.012	8052.081
TVAE	0.794	0.616	0.819	0.18	4699.311
CTABGAN+	0.741	0.482	0.713	0.061	5518.107
StaSy	0.793	0.602	0.808	$\ll 0$	1.06×10^8
TabDDPM	0.820	**0.665**	**0.852**	-9.39	17322.613
AutoGAN	0.618	0.356	0.647	$\ll 0$	2.47×10^{11}
Stasy-AutoDiff	0.834	0.620	0.846	**0.199**	4429.56
Med-AutoDiff	0.777	0.581	0.819	0.1	**4281.12**
Tab-AutoDiff	**0.836**	0.659	**0.852**	0.172	4679.11

Table 2: Classification/Regression Performance on Different Synthetic Data. We reported the average of accuracy, F1 score, AUROC for classification tasks, and R^2 and RMSE for regression tasks. Identity refers to the classifiers/regressors trained on the training set of the real data. Accuracy refers to the proportion of predictions that equal the real label.

Overall AutoDiff outperforms AutoGAN model in fidelity measures for all 5 metrics. We present the correlation plots of AutoDiff and TabDDPM for Abalone dataset in the Appendix C.

Utility Comparisons. We evaluated the efficacy of AutoDiff in preserving the Machine Learning Utility of synthesized data, following the Train on Synthetic and Test on Real (TSTR) [7] pipeline. For each real dataset, we split it into training/test set with the ratio of 80%/20%. Then we fit two prediction models, one on the training set of real data and one on the corresponding synthetic table. Finally we test the performance of both models on the test set of the real data. For classification tasks we used Naïve Bayes, K-Nearest Neighbors, Decision Tree, Random Forest, XGBoost, LightGBM, and CatBoost. For regression tasks we used Linear Regression, Lasso regression, Decision Tree, Random Forest. Table 2 illustrates the aggregate performance metrics averaged across all datasets, the performances of classifiers and regressors trained on synthetic tables generated by various synthesizers are reported. Notably, AutoDiff consistently outperforms GAN and Autoencoder-based methods, such as CTGAN, CTABGAN+, AutoGAN, and TVAE by a substantial margin. When compared to other diffusion-based methods, AutoDiff exhibits superior performance; it notably surpasses the original Stasy method that does not employ autoencoder-based encoding. While the improvement that AutoDiff demonstrates over the DDPM-based model (TabDDPM) is on-par for classification tasks, it is significantly more pronounced for regression tasks, as evidenced by notably higher R^2 values and lower RMSE scores. It is interesting to note the employment of gaussian quantile transformation and score-network structure from TabDDPM improves the performances in classification tasks when comparing Stasy- and Tab-Autodiffs over all three metrics.

Privacy Comparisons. Privacy is an important issue in synthetic tabular data generation, and we simply measure it through the Distances to Closest Records (DCR) in [25]. For each synthetic sample, DCR computes the minimum L2-distance to real records, and Mean DCR averages these distances over all synthetic samples. Essentially, low DCR indicates synthetic samples memorize some data points in real table, violating some privacy restrictions. In contrast, high DCR values denote the synthesizer generates some new data points, which cannot be observed in real-table. But note that some random noises can have high DCR values. Therefore, DCR should be considered with fidelity and utility together. In Table 3, the averaged rankings of Mean-DCR (MDCR) for 9 models are presented. High ranking indicates the model has low MDCR values. The reason why we adopt the ranking is due to the large variability of MDCR per dataset. We present the MDCR values per each dataset over 9 models in the Appendix D. The auto-encoder based methods, i.e., AutoDiff, AutoGAN, TVAE, have relatively higher ranks of MDCR, than non auto-encoder based methods such as Stasy or TabDDPM. This result is expected as sophisticatedly designed auto-encoder frequently overfits the input data, combined with the memorization issues of diffusion model [2]. We observe that the overall rank of

Models	MDCR Rank
CTGAN	4.33
TVAE	7.00
CTABGAN+	3.87
Stasy	1.93
TabDDPM	3.40
AutoGAN	4.93
Stasy-AutoDiff	6.13
Med-AutoDiff	**7.07**
Tab-AutoDiff	6.33

Table 3: Rank of MDCR over different models. The highest rank (resp. second to the highest) is bolded (resp. underlined) implying synthesized datasets are close to the real.

MDCR is similar to that of fidelity for numerical variables in Table 1, and strongly conjecture the relatively good performance of TabDPPM in MDCR measure over AutoDiff is attributed to the fact that it does not capture the correlation structures among features but shows good fidelity to individual feature.

Future works. We believe ensuring the privacy guarantees with good statistical fidelity and utility of synthesized tabular data is one of the most fundamental questions we should resolve. Under a rigorous framework of differential privacy (DP) [6], further extending the current form of AutoDiff model that satisfies the DP-constraints is the most promising directions for the future work [5, 16].

References

[1] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[2] Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270, 2023.

[3] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR, 2017.

[4] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[5] Tim Dockhorn, Tianshi Cao, Arash Vahdat, and Karsten Kreis. Differentially private diffusion models. *arXiv preprint arXiv:2210.09929*, 2022.

[6] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[7] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[9] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[10] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR, 2022.

[11] Jayoung Kim, Chaejeong Lee, and Noseong Park. Stasy: Score-based tabular data synthesis. *arXiv preprint arXiv:2210.04018*, 2022.

[12] Ron Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

[13] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. *arXiv preprint arXiv:2209.15421*, 2022.

[14] Chaejeong Lee, Jayoung Kim, and Noseong Park. Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis. *arXiv preprint arXiv:2304.12654*, 2023.

[15] Xingchao Liu, Lemeng Wu, Mao Ye, et al. Learning diffusion bridges on constrained domains. In *The Eleventh International Conference on Learning Representations*, 2022.

[16] Saiyue Lyu, Margarita Vinaroz, Michael F Liu, and Mijung Park. Differentially private latent diffusion models. *arXiv preprint arXiv:2305.15759*, 2023.

[17] Gustav Müller-Franzes, Jan Moritz Niehues, Firas Khader, Soroosh Tayebi Arasteh, Christoph Haarburger, Christiane Kuhl, Tianci Wang, Tianyu Han, Sven Nebelung, Jakob Nikolas Kather, et al. Diffusion probabilistic models beat gans on medical images. *arXiv preprint arXiv:2212.07501*, 2022.

[18] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[20] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[21] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[23] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[24] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32, 2019.

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

[26] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. Ctab-gan+: Enhancing tabular data synthesis. *arXiv preprint arXiv:2204.00401*, 2022.

A Real Tabular-data List

We provide the URL for the sources of each dataset considered in the paper. The provided information also includes the downstream task that each dataset is designed for, which may include binary classification, multi-class classification, or regression.

1. **abalone** (OpenML) : https://www.openml.org/search?type=data&sort=runs&id=183&status=active (Multi class)
2. **adult** (Kohavi, R) : Check the reference [12]. (Binary class)
3. **Bean** (UCI) : https://archive.ics.uci.edu/dataset/602/dry+bean+dataset (Multi class)
4. **Churn-Modelling** (Kaggle) : https://www.kaggle.com/datasets/shrutimechlearn/churn-modelling (Binary class)
5. **faults** (UCI) : https://archive.ics.uci.edu/dataset/198/steel+plates+faults (Multi class)
6. **HTRU** (UCI) : https://archive.ics.uci.edu/dataset/372/htru2 (Binary class)
7. **indian liver patient** (Kaggle) : https://www.kaggle.com/datasets/uciml/indian-liver-patient-records?resource=download (Binary class)
8. **insurance** (Kaggle) : https://www.kaggle.com/datasets/mirichoi0218/insurance (Regression)
9. **Magic** (Kaggle) : https://www.kaggle.com/datasets/abhinand05/magic-gamma-telescope-dataset?resource=download (Binary class)
10. **News** (UCI) : https://archive.ics.uci.edu/dataset/332/online+news+popularity (Regression)
11. **nursery** (Kaggle) : https://www.kaggle.com/datasets/heitornunes/nursery (Multi class)
12. **Obesity** (Kaggle) : https://www.kaggle.com/datasets/tathagatbanerjee/obesity-dataset-uci-ml (Multi class)
13. **Shoppers** (Kaggle) : https://www.kaggle.com/datasets/henrysue/online-shoppers-intention (Binary class)
14. **Titanic** (Kaggle) : https://www.kaggle.com/c/titanic/data (Multi class)
15. **wilt** (OpenML) : https://www.openml.org/search?type=data&sort=runs&id=40983&status=active (Binary class)

B Score-based diffusion model

Diffusion model consists of two processes. The first step, *forward process*, transforms data into noise. Specifically, the score-based diffusion model [23] uses the following stochastic differential equations (SDEs) [21] for data perturbation:

$$d\mathbf{x} = f(\mathbf{x}, t)dt + g(t)dW_t, \tag{4}$$

where $f(\mathbf{x}, t)$ and $g(t)$ are drift and diffusion coefficient, respectively, and W_t is a standard Wiener-process (a.k.a. Brownian Motion) indexed by time $t \in [0, T]$. Here, the f and g functions are user-specified, and [23] suggests three different types of SDEs, i.e., variance exploding (VE), variance preserving (VP), and sub-variance preserving (sub-VP) SDEs for data perturbation. In this paper, we used VP-SDE defined as :

$$d\mathbf{x} = -\frac{1}{2}\beta(t)\mathbf{x}dt + \sqrt{\beta(t)}dW_t, \tag{5}$$

where $\beta(t) := \beta(0) + (\beta(1) - \beta(0))t$ with $\beta(1) := 20$ and $\beta(0) := 0.1$.

The second step, *reverse process*, is a generative process that reverses the effect of the forward process. This process learns to transform the noise back into data by reversing SDEs in (4). Through the Fokker-Planck equation of marginal density $p_t(\mathbf{x})$ of (4) for each fixed time $t \in [0, T]$, the following reverse SDE [1] can be easily derived:

$$d\mathbf{x} = \Big[f(\mathbf{x}, t) - g(t)^2\nabla_{\mathbf{x}} \log p_t(\mathbf{x})\Big]dt + g(t)d\bar{W}_t. \tag{6}$$

Here, the gradient of $\log p_t(\mathbf{x})$ w.r.t to the perturbed data $\mathbf{x}(t)$ is referred to as score function, dt in (6) is an infinitesimal negative time step, and $d\bar{W}_t$ is a Wiener-Process running backward in time, i.e., $t : T \to 0$.

The score function, $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, is approximated by a time-dependent score-based model $\mathbf{S}_\theta(\mathbf{x}(t), t)$ which is parametrized by neural networks in practice. The network parameter θ is estimated through minimizing the score-matching loss:

$$\theta^\star := \arg\min_\theta \mathbb{E}_{t\sim\mathcal{U}[\varepsilon,T]}\mathbb{E}_{\mathbf{x}(\mathbf{t})|\mathbf{x}(\mathbf{0})}\mathbb{E}_{\mathbf{x}(0)}\left[\lambda(t)^2 \left\|\mathbf{S}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) \mid \mathbf{x}(0))\right\|_2^2\right], \quad (7)$$

where $\mathcal{U}[\varepsilon, T]$ is an uniform distribution over $[\varepsilon, T]$, and $\lambda(t)(> 0)$ is a positive weighting function that helps the scales of matching-losses $\|\mathbf{S}_\theta(\mathbf{x}(t), t) - \nabla_{\mathbf{x}} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))\|_2^2$ to be in the same order across over the time $t \in [\varepsilon, T]$. The ε is set up to prevent the divergence of score function when t approached 0, specifically when ground-truth $p_0(\mathbf{x})$ is non-smooth. We set $\varepsilon := 1e^{-5}$ and $T = 1$ throughout the paper. The weight function is often chosen as $\lambda(t) \propto 1/\sqrt{\mathbb{E}\|\left[\nabla_{\mathbf{x}} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))\right]\|_2^2}$. We set $\lambda(t) := 1/\sigma(t)$, where $\sigma(t)$ is a standard deviation of $\mathbf{x}(t) \mid \mathbf{x}(0)$: $\sigma(t) := 1 - \exp\big(-\frac{1}{2}\int_0^t \beta(s)ds\big)$. See Appendix B in [23].

Score network of Stasy-AutoDiff. Let d be the dimension of $\mathbf{x}^{\text{Lat}}$ and the model is run over $t \in [10^{-5}, 1]$. Under this setting, the score network is constructed as follows:

$$\begin{aligned}
&\mathbf{h}_0 = \mathbf{x}^{\text{Lat}}(t) \in \mathbb{R}^d,\\
&\mathbf{h}_i = \text{ELU}(\mathbf{H}_i(\mathbf{h}_{i-1}, t) \oplus \mathbf{h}_{i-1}), \quad 1 \le i \le d_N,\\
&S_\theta(\mathbf{x}^{\text{Lat}}(t), t) = \text{Linear}(\mathbf{h}_{d_N}) \in \mathbb{R}^d,
\end{aligned}$$

where $\mathbf{H}_i(\mathbf{h}_{i-1}, t)$ is the concat-squash function [9] defined as

$$\mathbf{H}_i(\mathbf{h}_{i-1}, t) := \text{Linear}_i(\mathbf{h}_{i-1}) \odot \phi(\text{Linear}_i^{\text{gate}}(t) + \text{Linear}_i^{\text{bias}}(t)).$$

Here, $\odot$ is an element-wise multiplication, $\oplus$ means the concatenation operator, and ϕ is a sigmoid function. Throughout this paper, we set $(d_1, d_2, d_3, d_4, d_5) = (256, 512, 1024, 512, 256)$.

Score network of Tab-AutoDiff. Denote $t \in [10^{-5}, 1]$ as a timestep, and SinTimeEmb refers to a sinusoidal time embedding as in [18] with dimension of 128. For any fixed $t \in [10^{-5}, 1]$, time embedding *t-emb* and the input of the score-network $\mathbf{x}(t)$ is given as :

$$\begin{aligned}
&\textit{t-emb} = \text{Linear}(\text{SiLU}(\text{Linear}(\text{SinTimeEmb(t)}))),\\
&\mathbf{x}(t) = \text{Linear}(\mathbf{x}^{\text{Lat}}(t)) + \textit{t-emb}.
\end{aligned}$$

All Linear layers in (11) and (12) have a fixed projection dimension 128. Then, we have the final output of time-dependent score network can be computed as follows:

$$\begin{aligned}
&\text{MLPBlock}(\mathbf{x}) = \text{Dropout}(\text{ReLU}(\text{Linear}(\mathbf{x}))),\\
&S_\theta(\mathbf{x}^{\text{Lat}}(t), t) = \text{Linear}(\text{MLPBlock}(\cdots \text{MLPBlock}(\mathbf{x}(t)))).
\end{aligned}$$

In this paper, we used 4 blocks of MLP.

Note that in Stasy-AutoDiff, min-max scaler is used for processing the numerical variables, whereas in Tab-AutoDiff, Gaussian Quantile Transformation is used for the processing of numerical variables.

C Correlation Plots of Abalone

We provide the correlation plots of abalone dataset of synthetic tables from Stasy-AutoDiff and TabDDPM. The heatmaps on the top of Figures 3 and 4 display the difference of correlation plots from real and synthetic datasets. We provide more correlation plots from models considered in the paper in

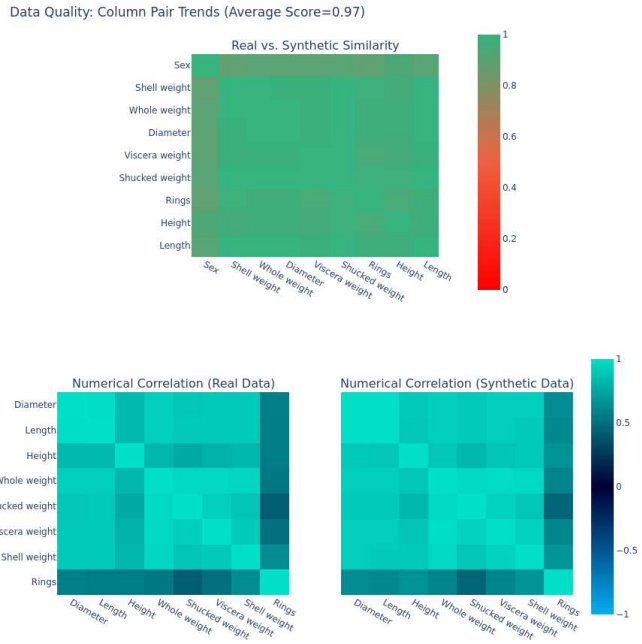

Figure 3: Correaltion Heatmap of *"Abalone"* dataset from Stasy-AutoDiff.

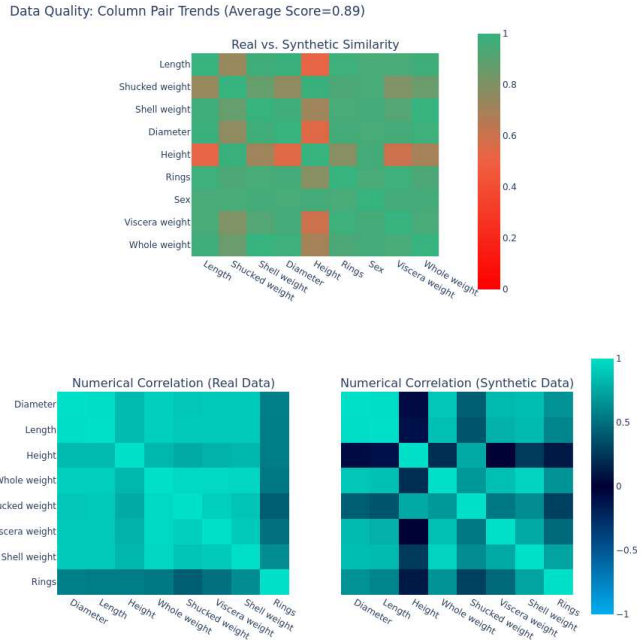

Figure 4: Correaltion Heatmap of *"Abalone"* dataset from TabDDPM.

D DCR values for each dataset

We provide the Mean-DCR values averaged over 10 synthetic datasets for each dataset. The smallest (resp. second to the smallest) is colored in black (resp. underlined), which implies synthesized datasets are close to the real.

	Titanic	**adult**	**HTRU**	**Magic**	**Bean**	**wilt**	**abalone**
CTGAN	5.24	109.60	10.63	29.64	9953.02	25.62	0.19
TVAE	**2.24**	**61.10**	6.93	**14.54**	2364.37	12.43	0.08
CTABGAN+	4.74	247.74	16.65	24.30	3778.49	17.35	0.16
Stasy	34.36	13150.87	94.45	82.08	55911.20	345.85	0.56
TabDDPM	4.63	126777.67	**3.22**	16.94	**130.32**	21.56	0.15
AutoGAN	2.04	720.72	5.70	24.28	645.77	12.13	0.09
Stasy-AutoDiff	2.89	195.92	4.53	16.95	432.73	**9.66**	0.08
Med-AutoDiff	2.53	282.34	3.98	18.97	634.65	10.14	0.12
Tab-AutoDiff	3.18	107.58	3.77	18.37	196.68	13.87	**0.07**

	Patient	**nursery**	**faults**	**Obesity**	**News**	**INSR**	**CM**	**SHP**
CTGAN	46.25	0.28	1148213	3.79	11587.43	35.00	693.81	30.08
TVAE	**20.90**	0.39	496487.4	3.38	**6736.04**	29.03	**590.43**	19.38
CTABGAN+	106.62	0.23	1499634	3.10	24340.9	48.93	749.84	30.50
Stasy	81.29	**0.22**	3327054	4.86	2.71×10^8	1106.14	7017.03	4824.17
TabDDPM	256.59	0.95	4137191	98.82	705767.7	50.29	1699.10	9316.37
AutoGAN	46.80	1.01	300791.7	3.47	40104.49	35.47	5851.55	**17.10**
Stasy-AutoDiff	33.59	0.40	**152907.3**	**2.25**	15147.05	**27.32**	707.86	27.83
Med-AutoDiff	31.11	0.56	68837.95	2.69	20369.19	30.65	896.85	28.32
Tab-AutoDiff	75.77	**0.22**	278249.9	2.65	37245.33	38.22	688.58	64.98