CG-TGAN: Conditional Generative Adversarial Networks with Graph Neural Networks for Tabular Data Synthesizing

Seungcheol Lee*, **Moohong Min**

Sungkyunkwan University
cheol11111@g.skku.edu, iceo@skku.edu

Abstract

Data sharing is necessary for AI to be widely used, but sharing sensitive data with others with privacy is risky. To solve these problems, it is necessary to synthesize realistic tabular data. In many cases, tabular data contains a mixture of continuous, mixed, categorical columns. Moreover, columns of the same type may have multimodal distribution or be highly imbalanced. These issues make it challenging to synthesize tabular data. The synthesized tabular data should reflect the relational meaning between columns of tabular data, so modeling the probability distribution of tabular data is a non-trivial task. Traditional tabular data synthesizing models are based on GAN or diffusion models and are built using fully connected or convolutional layers. However, fully connected layers have the disadvantage of low inductive bias, and convolutional layers are not invariant to the column order of tabular data. Therefore, we assume that converting tabular data into graph-structured data and using a graph neural network would produce better synthetic data than using fully connected layers or convolutional layers. Our study aims to show that GANs constructed with graph neural networks can outperform existing GAN models using fully connected layers or convolutional layers. We propose CG-TGAN, a conditional GAN built using graph neural networks. To learn how to synthesize realistic data, the graph neural networks in the discriminator and generator learn graph-level tasks and node-level tasks together. The discriminator of CG-TGAN learns a graph-level task to distinguish between real and synthetic data and node-level tasks to predict the value of the target node. CG-TGAN's generator learns a graph-level task to synthesize an overall graph similar to real data and node-level tasks to learn how to synthesize a fake graph with the proper relation between nodes. In this paper, we show that CG-TGAN outperforms GAN-based models and is comparable to diffusion-based models.

Introduction

For AI to be applied in many fields, data sharing is necessary. Tabular data is widely used in many tasks, such as online advertising, recommendation, fraud detection, medical treatment, etc.(Luo et al. 2020) However, sharing sensitive data that contains private information or corporate security is risky. To address this problem, generating realistic synthetic data can significantly benefit many AI applications by making data sharing safer and more accessible.

The mainstream of existing research in tabular data synthesizing is using GAN(Goodfellow et al. 2014) or diffusion(Ho, Jain, and Abbeel 2020) models. The research using diffusion model(Kotelnikov et al. 2023; Lee, Kim, and Park 2023) uses fully connected layers for constructing a tabular data synthesizing model. The models using GAN can be categorized by which layers are used to construct the generator and discriminator of GAN. In previous studies(Park et al. 2018; Zhao et al. 2021, 2022), the rationale for using convolutional layers was that local features could be extracted better than fully connected layers. However, when using convolutional layers, the order of tabular data columns can change the results, even though the order of columns is not meaningful. Moreover, the difficulty of extracting combined features between different columns depends on the closeness of column orders. In other words, the results of extracting features from tabular data using convolutional layers are not invariant from the order of columns. We wanted to find a better and more intuitive way to learn relationships between tabular data columns.

We started our research with the hypothesis that "if we convert tabular data to graph-structured data and construct the tabular data synthesizer model with graph neural networks(GNN), we could generate better synthetic data." Our hypothesis was inspired by the Structural Causal Model(SCM)(Pearl 2009). A SCM $\mathcal{M}$ can be represented with a triplet $< \mathbf{U}, \mathbf{V}, \mathcal{F} >$. $\mathbf{U}$ is a set of exogenous variables, $\mathbf{V}$ is a set of endogenous variables, and $\mathcal{F} = \{f_1, f_2, \cdots, f_n\}$ is a set of functions which are mapping from $\mathbf{U}$ to $\mathbf{V}$. Using SCM, if we know the parent nodes of node i we can assign a value to v_i like as $v_i \leftarrow f_i(pa_i, u_i)$. Unfortunately, the causal relationships between tabular data columns are not generally known, and learning them is very difficult. Therefore, it is impractical to implement a generative model with a probabilistic graph model such as Bayesian networks because they rarely meet the prerequisite of having a causal graph of tabular data. However, we are interested in the significant context of being able to generate data from a graph, which led us to the above hypothesis.

To prove our hypothesis, we propose CG-TGAN, a conditional GAN(CGAN)(Mirza and Osindero 2014) that uses

*Corresponding author

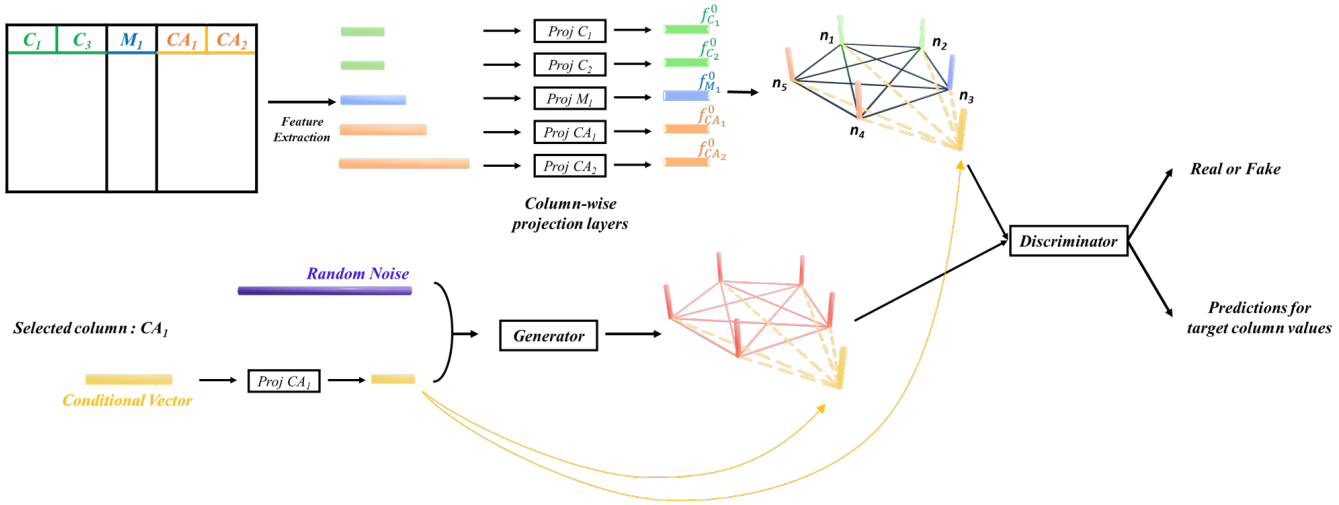

Figure 1: Overall CG-TGAN workflow. To convert real tabular data to graph, we extracted features from each column and passed them through a projection layer for each column to get $f^0_{C_1}$, $f^0_{C_2}$, $f^0_{M_1}$, $f^0_{CA_1}$, $f^0_{CA_2}$ vectors with the same length. In the figure, a one-hot conditional vector generated from the categorical column (CA_1) is passed through the projection layer corresponding to the CA_1 column and used as a conditional vector for the generator's input and the conditional node's vector in the real and fake graph. The generator synthesizes a fake graph from a conditional vector and random noise. Real and fake graphs with a conditional node are input to the discriminator. And the discriminator learns a graph-level task to determine whether they are real or fake, and a node-level task to predict the value of the target node.

GNN to construct a generator and discriminator. We chose a GAN-based model because there are more prior studies than diffusion models; It is a good way to compare the utility of our ideas. One of the main contents we want to show in this paper is that CG-TGAN can outperform other GAN-based models. We convert tabular data into a fully connected graph with weighted edges to use GNN on tabular data. The converted graph($\mathcal{G} = (\mathcal{V}, \mathcal{E})$)'s nodes($v_i \in \mathcal{V}$) correspond to columns in the tabular data, and edges($(v_i, v_j) \in \mathcal{E}$) correspond to relationship between columns. We bypassed the problem of not being given relationship information between columns of tabular data by using a learnable parameter matrix(A_P). We need two adjacency matrices. A_G is for generator and A_D is for discriminator. A_G and A_D are calculated from A_P and A_P is trained together during the CG-TGAN training process.

In the training process, the GNN in the discriminator and generator learn graph-level tasks and node-level tasks. For the discriminator, we trained it with a graph-level task to determine whether the input graph is real or fake and a node-level task to predict the value of the target node. We let the generator learn a graph-level task, how to synthesize an overall graph similar to real data, and a node-level task to produce a graph that captures the relationships between nodes. We use Graph Convolutional Networks(GCN)(Kipf and Welling 2016) to construct the generator and discriminator in CG-TGAN.

We applied CG-TGAN to seven datasets and compared our model with existing tabular data synthesizing models, such as DPGAN(Xie et al. 2018), Pate-GAN(Jordon, Yoon, and Van Der Schaar 2018), CTGAN(Xu et al. 2019), CTAB-GAN(Zhao et al. 2021), CTAB-GAN+(Zhao et al. 2022), CoDi(Lee, Kim, and Park 2023), and TabDDPM(Kotelnikov et al. 2023). Our CG-TGAN outperforms existing GAN-based models. Also, compared to diffusion-based models, CG-TGAN outperforms CoDi. And it is comparable to TabDDPM in terms of machine learning utility and performs better in privacy protection, even though diffusion models generally outperform GAN-based models(Dhariwal and Nichol 2021).

Our main contributions are as follows:

- We propose a novel method for tabular data synthesis by constructing CGAN using GNN.
- Our CG-TGAN shows a better performance than existing GAN-based models.
- We propose the training process of CG-TGAN, which learns graph-level and node-level tasks sequentially. By enabling CG-TGAN to learn node-level tasks that predict the value of the target node, we enhance its ability to generate synthetic data that accurately reflects the intricate relationships between nodes.
- We propose the conditional node concept for constructing CGAN with GNN. We provide the conditional vector input to the discriminator through the conditional node.

Related Works

Tabular Data Synthesizers

Existing tabular data synthesizer models can be classified according to the type of layers used to construct the model. The models using fully connected layers are CoDi(Lee, Kim, and Park 2023), TabDDPM(Kotelnikov et al. 2023), CTGAN(Xu et al. 2019), medGAN(Choi et al. 2017),

CrGAN-Cnet(Mottini, Lheritier, and Acuna-Agost 2018), Pate-GAN(Jordon, Yoon, and Van Der Schaar 2018), and DPGAN(Xie et al. 2018). The models using convolutional layers are TableGAN(Park et al. 2018), CTAB-GAN(Zhao et al. 2021), and CTAB-GAN+(Zhao et al. 2022). The disadvantage of using fully connected layers is weak inductive bias. And the models using convolutional layers suffer from the problem of not being invariant to tabular data's column order.

Many tabular data synthesizers(Xu et al. 2019; Engelmann and Lessmann 2021; Zhao et al. 2021, 2022) use CGAN. Because CGAN can increase the learning power for imbalanced datasets by providing conditional vectors to ensure that the generated data belongs to a particular minority class. Some studies use CGAN with auxiliary classifier(Odena, Olah, and Shlens 2017) to improve the performance of tabular data synthesizers. TableGAN(Park et al. 2018) and CTAB-GAN(Zhao et al. 2021) use an auxiliary classifier. Although the discriminator can learn the semantic integrity of tabular data by itself as the discriminator is not mainly focused on learning semantic integrity, the auxiliary classifier is used with the expectation that the auxiliary classifier will provide feedback to the generator to generate fake data with a more appropriate relationship between columns.

CG-TGAN's discriminator and generator are designed to learn relations between tabular data columns by using GNN. In CG-TGAN, we give the discriminator the additional role of predicting the target node value. To do this, the GNN in the discriminator is additionally trained on node-level tasks.

Graph Neural Networks

A graph neural network is an artificial neural network for processing graph-structured data. It extracts features from graph-structured data and uses them to perform downstream tasks. The GNN layer can be described in three steps by three functions (message(ϕ), aggregation(α), update(γ)).(Gilmer et al. 2017)

1. Each node in the graph receives some messages from its neighbors.
2. Each node in the graph aggregates the messages from its neighbors.
3. Each node in the graph updates its node feature using the aggregated messages and the original node feature.

The above process can be expressed as Equation 1.

$$\mathbf{f}_i^{t+1} = \gamma(\mathbf{f}_i^t, \alpha_{j \in \mathcal{N}(i)}(\phi(\mathbf{f}_i^t, \mathbf{f}_j^t))) \tag{1}$$

There are several variants of GNN depending on which message(ϕ), aggregation(α), and update(γ) functions are used. Graph Convolution Networks(GCN)(Kipf and Welling 2016) is a method that bridges spectral and spatial methods, using the property that the convolution operation in the spatial domain is equivalent to a product in the Fourier domain. GCN uses the convolution operation as an Aggregate function. GraphSAGE(Hamilton, Ying, and Leskovec 2017) uses concatenated vectors of self and neighbor embeddings for aggregation and introduces Mean, Pooling, and LSTM aggregators. GAT(Velickovic et al. 2017) does not consider each neighbor to be equally important, but instead applies attention(Vaswani et al. 2017) coefficients to give more weight to essential nodes.

We use GCN to organize discriminators and generators. GCN is simple to implement and has the advantage of being a baseline model for many tasks.

Methodology

Technical Background

Data Encoding Methods for encoding tabular data have been studied in previous research. Mode-specific normalization method(Xu et al. 2019) is proposed to handle continuous columns with multi-mode distribution, and mixed-type encoder(Zhao et al. 2021) is proposed to handle mixed-type columns which contain both categorical and continuous attributes. An example of a mixed-type variable is the capital gain of the Adult dataset. Capital gain refers to an increase in a capital asset when sold. The variable has a zero value for most people without capital assets because they do not have assets to sold and a non-zero value for the few who do have capital assets. Therefore, the capital gain variable is not a simple continuous variable but a variable with a categorical attribute.

Mode-specific normalization(Xu et al. 2019) is trained using Gaussian Mixture Model(Bishop 2006) to represent a continuous column distribution using m Gaussian distributions. From the trained m Gaussian distributions, we can calculate the probability that each data point belongs to each Gaussian distribution and find the Gaussian distribution with the maximum probability of belonging. A vector β can be constructed as in Equation 3 where only the element corresponding to the kth Gaussian distribution with the maximum probability has a value of 1, and all other values have a value of 0.

$$\alpha = \frac{c_{i,j} - \mu_k}{4\sigma_k} \tag{2}$$

$$\beta = [\beta_1, \beta_2, \cdots, \beta_m],\ \beta_i \in \{0, 1\} \tag{3}$$

The value of the ith row of the jth continuous column, $c_{i,j}$, can be represented as $\alpha \bigoplus \beta$ by concatenating scalar value α, which is the standardization of $c_{i,j}$ using kth Gaussian distribution's mean and standard deviation as shown in Equation 2 and one-hot vector β in Equation 3.

$$\beta = [\beta_1, \beta_2, \cdots, \beta_m; \beta_{c_1}, \cdots, \beta_{c_t}],\ \beta_i \in \{0, 1\} \tag{4}$$

A mixed-type encoder(Zhao et al. 2021) adds a categorical attribute one-hot encoding to mode-specific normalization as shown in Equation 4. A value equal to a categorical attribute item is represented with a one-hot vector $(\beta_{c_1}, \cdots, \beta_{c_t})$ with a value of 1 for only the elements representing that categorical item and a scalar α value of 1. The values not equal to categorical items are represented in the same way as mode-specific normalization.

We adopted mode-specific normalization for normal continuous column encoding and mixed-type encoder for mixed column encoding and used one-hot encoding for categorical column encoding.

CG-TGAN Structure

Tabular Data to Graph-Structured Data We first need to extract meaningful features from each column to convert tabular data into graph-structured data. To extract meaningful features for each column of tabular data, we use different encoders for each column type. For the continuous columns, we used mode-specific normalization. For the mixed-type columns, we used mixed-type encoder. For the categorical columns, we used one hot encoding. After extracting features from each column, to fit the extracted features with the same dimension($\mathbb{R}^d$) to construct the node feature matrix, we use projection layers, as shown in Figure 1.

$$\mathbf{f}^0 = \{\{\mathbf{f}^0_{C_i}\}_{i=1}^{N_c}; \{\mathbf{f}^0_{M_i}\}_{i=1}^{N_m}; \{\mathbf{f}^0_{CA_i}\}_{i=1}^{N_{ca}}\} \tag{5}$$

$$\mathbf{f}^0 = \{\{\mathbf{f}^0_{C_i}\}_{i=1}^{N_c}; \{\mathbf{f}^0_{M_i}\}_{i=1}^{N_m}; \{\mathbf{f}^0_{CA_i}\}_{i=1}^{N_{ca}}; \mathbf{f}_{cond}\} \tag{6}$$

The set of node features which is extracted from the tabular data with N_c continuous columns, N_m mixed columns, and N_{ca} categorical columns is shown in Equation 5. The set with the conditional node feature is shown in Equation 6. The input graph to the generator's GNN consists of the set(Equation 5), i.e., no conditional node is included in the input graph. On the other hand, the input graph to the discriminator's GNN consists of the set(Equation 6), i.e., the conditional node is included in the input graph. $\mathbf{f}^0$ is the set of node feature vectors in the input graph before passing through the GNN, and $\mathbf{f}^t$ is the set of node features in the graph after passing through the tth layer of the GNN.

Using the node features $\mathbf{f}^0$, we constructed a fully connected graph($\mathcal{G} = (\mathcal{N}, \mathcal{E})$), where each node $n_i \in \mathcal{N}$ correspond to each column and edges corresponds to the interaction between columns. It is similar to Fi-GNN(Li et al. 2019) in that it constructs a feature graph with features extracted from each column. However, unlike Fi-GNN, which uses datasets consisting of only categorical columns and defining the adjacency matrix with attentional edge weights, we deal with datasets with a mixture of columns of various types and treat two adjacency matrices(A_G, A_D) for each generator and discriminator. To define A_G and A_D, we use the parameter matrix A_P. A_P is a learnable parameter matrix, and we initialize the parameter matrix A_P as one matrix(J_{N+1}) minus identity matrix(I_{N+1}), as shown in Equation 8. This allows us to initialize the parameter matrix A_P, that each node is fully connected to the other with the same weight. A_G is extracted from A_P by slicing the region excluding the conditional node, as shown in Equation 9. A_D is the inverse matrix of A_P, as shown in Equation 10. The process of a generator producing data from random noise and the process of a discriminator mapping the data into feature space can be considered invertible.

$$N = N_c + N_m + N_{ca} \tag{7}$$

$$A_P = \frac{1}{\sqrt{N+1}}(J_{N+1} - I_{N+1}) \in \mathbb{R}^{N+1 \times N+1} \tag{8}$$

$$A_G = (A_{Pij})_{1 \le i \le N, 1 \le j \le N} \in \mathbb{R}^{N \times N} \tag{9}$$

$$A_D = A_P^{-1} \in \mathbb{R}^{N+1 \times N+1} \tag{10}$$

Conditional Vector and Node To configure a conditional GAN, we must put not only real or fake data but also a conditional vector into the discriminator. We propose creating a conditional node at the input graph to put a conditional vector in the discriminator. Conditional node is inspired by the concept of virtual nodes. The virtual node improves the performance of graph classification(Hwang et al. 2022) because the virtual node can help aggregate the representations of the entire graph and provide shortcuts for message passing. We created a conditional node fully connected to every graph node to pass the conditional vector information to all nodes. The revised graph by adding a conditional node is shown in Figure 1. The node features of a graph with a conditional node can be represented by Equation 6.

Generator and Discriminator We use graph convolutional layers(Kipf and Welling 2016) to construct GNNs for the CG-TGAN generator and discriminator. They are basically composed of three graph convolutional layers. We use the residual connections(He et al. 2016) to build GNNs.

The discriminator learns graph-level and node-level task. Both tasks share the graph convolutional layers. The purpose of learning the graph-level task is to learn the ability to determine whether the input is real or fake: it is the same as the purpose of general GAN's discriminator. The node-level task aims to teach the relationships between tabular data columns. By learning the relationships between tabular data columns, the discriminator can provide more accurate feedback to the generator so that if the value of synthesized data's target column is not well related to other columns, the generator no longer produces these data. Of course, the discriminator can learn the relationships between columns without node-level task, but this is not perfect, so we need to teach the discriminator node-level task.

The generator takes a random noise and a conditional vector as input. The generator first passes the concatenated vector of random noise and a conditional vector through a fully connected layer to generate a vector the size of the embedding dimension multiplied by the number of columns. The vector is then cut by the number of columns to create the node feature set as shown in Equation 5. Then it is passed through GNN to embed fake graph. The generator learns the graph-level and node-level task. The purpose of the graph-level task is to learn how to synthesize a fake graph that is macroscopically similar to the real graph. The node-level task aims to learn how to synthesize a fake graph with the correct relations between other nodes.

CG-TGAN Training

CG-TGAN training proceeds in the order of graph-level discriminator training, node-level discriminator training, graph-level generator training, and node-level generator training during a number of updates. The node-level task was trained with target column in each dataset. To train the graph-level task, we use wgan-gp(Gulrajani et al. 2017) loss. To train node-level task, we use cross-entropy loss for categorical column and mean squared error loss for continuous or mixed column. We use Adam(Kingma and Ba 2014) optimizer with learning rate 1e-4.

Algorithm 1: CG-TGAN Training. All experiments in the paper used the default values $\alpha = 0.0001$, $n_{dg} = 5$, $n_{update} = 5000$, $b = 256$

Require: α, learning rate. n_{dg}, the number of iterations of the discriminator's graph-level task per update. n_{update}, the number of updates for CG-TGAN learning. b, the batch size. c_{idx}, the index of target column.
Require: ω_0, initial discriminator parameters. θ_0, initial generator's parameters. δ_0, initial classifier's parameters. ϕ_0, initial adjacency matrix's parameters.
Notations: f_ω, discriminator function. g_θ, generator function. c_δ, classifier function. L_{dwg}, discriminator WGAN-GP loss function. L_{gwg}, generator WGAN-GP loss function. L_{pred}, MSE or CSE loss function.
for updates $1, \cdots, n_{update}$ **do**
  Sample $\{x^{(i)}\}_{i=1}^b$, $\{y^{(i)}\}_{i=1}^b$ from real data.
  (Discriminator Graph-level Task)
  for steps $1, \cdots, n_{dg}$ **do**
    Sample $\{z^{(i)}\}_{i=1}^b$ from random noise.
    $Loss_{dg} \leftarrow L_{dwg}(\{x^{(i)}\}_{i=1}^b, \{g_\theta(z^{(i)})\}_{i=1}^b)$
    $\omega \leftarrow \text{Adam}(\nabla_\omega Loss_{dg})$
    $\phi \leftarrow \text{Adam}(\nabla_\phi Loss_{dg})$
  end for
  (Discriminator Node-level Task)
  $Loss_{dn} \leftarrow \sum_{i=1}^b \frac{1}{b} L_{pred}(c_\delta(f_\omega(x^{(i)})), y^{(i)})$
  $\omega \leftarrow \text{Adam}(\nabla_\omega Loss_{dn})$
  $\phi \leftarrow \text{Adam}(\nabla_\phi Loss_{dn})$
  $\delta \leftarrow \text{Adam}(\nabla_\delta Loss_{dn})$
  (Generator Graph-level Task)
  Sample $\{z^{(i)}\}_{i=1}^b$ from random noise.
  $Loss_{gg} \leftarrow L_{gwg}(\{x^{(i)}\}_{i=1}^b, \{g_\theta(z^{(i)})\}_{i=1}^b)$
  $\theta \leftarrow \text{Adam}(\nabla_\theta Loss_{gg})$
  $\phi \leftarrow \text{Adam}(\nabla_\phi Loss_{gg})$
  (Generator Node-level Task)
  Sample $\{z^{(i)}\}_{i=1}^b$ from random noise.
  $\{\tilde{y}^{(i)}\}_{i=1}^b \leftarrow \{g_\theta(z^{(i)})[:, c_{idx}]\}_{i=1}^b$
  $Loss_{gn} \leftarrow \sum_{i=1}^b \frac{1}{b} L_{pred}(c_\delta(f_\omega(g_\theta(z^{(i)}))), \{\tilde{y}^{(i)}\}_{i=1}^b)$
  $\theta \leftarrow \text{Adam}(\nabla_\theta Loss_{gn})$
  $\phi \leftarrow \text{Adam}(\nabla_\phi Loss_{gn})$
end for

Experiments

Experimental Setup

Datasets To evaluate CG-TGAN, we selected seven widely used datasets. For regression, we used Abalone(Nash and Ford 1995), King(HARLFOXEM 2017), Insurance(Choi 2018) datasets. For binary classification, we used Adult(Kohavi et al. 1996), Diabetes(Sigillito 2014), Wilt(Johnson 2014) datasets. For multi-class classification, we used Gesture(Madeo, Lima, and Peres 2013) dataset.

Baselines To evaluate the performance of CG-TGAN, we compared it to five existing GAN-based tabular data synthesizers, DPGAN, Pate-GAN, CTGAN, CTAB-GAN and CTAB-GAN+. And we also compare with diffusion-based tabular data synthesizers, CoDi and TabDDPM.

Evaluation Methods

We compared the tabular data synthesizers in terms of machine learning utility and privacy protection.

The machine learning utility evaluation method compares the difference in performance between a machine learning model trained with real data and a model trained with synthetic data. The smaller the performance difference, the better the performance of the synthesizer. We trained logistic regression, decision tree(Quinlan 1986), and random forest(Breiman 2001) models for classification datasets and measured accuracy, the area under the ROC(AUC), and the F1 score. We trained linear regression, lasso(Tibshirani 1996), and ridge(Hoerl and Kennard 1970) regression models for regression datasets and measured mean absolute percentage error(MAPE)(De Myttenaere et al. 2016), mean absolute error(MAE), and R^2.

To compare privacy protection ability, we use the concept of Nearest Neighbor Distance Ratio(NNDR)(Zhao et al. 2021). Let $\mathbf{D}_i$ be a set of all distances between i^{th} synthetic data point and all real data points. $d_1 = min(\mathbf{D}_i)$ is the closest distance and $d_2 = min(\mathbf{D}_i \setminus d_1)$ is the second closest distance. NNDR $= \frac{d_2}{d_1}$ is a ratio; it is between 0 and 1. The closer it is to 1, the better the synthetic data point is located in the middle point between the real data points, and the safer it is from privacy risks. Closer to 0, the synthetic data strongly reflects the values of nearby real data, which can lead to a privacy breach.

Results

Comparison with GAN-based models

Comparison of Synthetic Data Distributions Figure 2 shows the distribution of synthesized Diabetes datasets by GAN-based models and CG-TGAN. It shows that the data generated by CG-TGAN has a closer distribution to the real data than the comparison models in all columns. Figure 3 shows the results of t-SNE(Van der Maaten and Hinton 2008) projection for synthesized Diabetes datasets by GAN-based models and CG-TGAN. DPGAN and Pate-GAN show mode collapsed results. CTGAN, CTAB-GAN, and CTAB-GAN+ fail to depict some regions of the real data. On the other hand, the data synthesized with CG-TGAN provides a comprehensive depiction, covering all regions of the real data.

Machine Learning Utility Table 1 and 2 show the machine learning utility evaluation results. The values in the tables show the difference in performance between the model trained on real data and the model trained on synthesized data. For MAPE and MAE, smaller is better, so the larger value in the table means the better performance of the synthesizer. For R^2, accuracy, AUC, and F1 score, larger is better, so the smaller value in the table means the synthesizer performs better. Compared to other GAN-based models, CG-TGAN produces better-synthesized data on all regression datasets. We also show that CG-TGAN outperforms other GAN-based models on classification datasets. The results in Table 1 and 2 show that CG-TGAN consistently outperforms other GAN-based models.

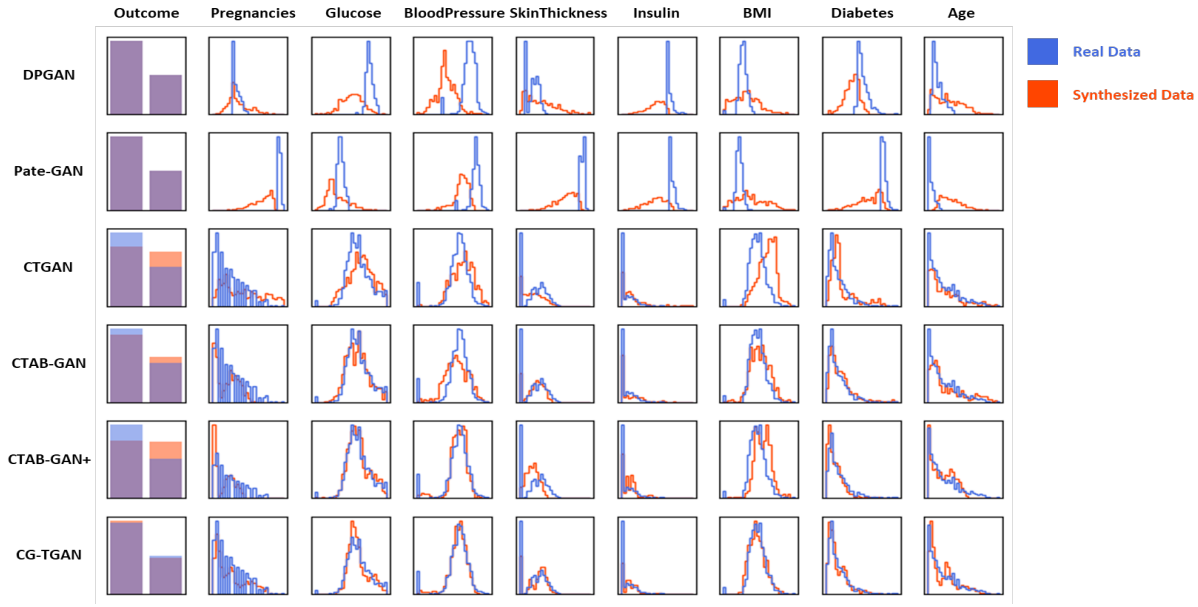

Figure 2: Comparison of synthetic data histograms on Diabetes dataset.

	Abalone dataset			Insurance dataset			King dataset		
	MAPE	MAE	R^2	MAPE	MAE	R^2	MAPE	MAE	R^2
DPGAN	F	F	F	F	F	F	F	F	F
Pate-GAN	F	F	F	F	F	F	F	F	F
CTGAN	-0.0111	-0.1765	0.1891	-0.9633	-4.421e+3	0.8240	-0.1222	-5.535e+5	0.2033
CTAB-GAN	-	-	-	-	-	-	-	-	-
CTAB-GAN+	-0.0257	-0.2309	0.1373	-0.9306	-3.381e+3	0.5276	-0.1045	-4.005e+5	0.1494
CG-TGAN	**-0.0046**	**-0.1174**	**0.1274**	**0.0489**	**1.115e+2**	**0.0065**	**-0.0006**	**-9.325e+3**	**0.1281**

Table 1: Machine Learning Utility Results on Regression Datasets. ”F” means that the quality of the synthesized data is too low to evaluate, and ”-” means that the tabular data synthesizer cannot generate regression datasets.

	Adult dataset			Diabetes dataset			Gesture dataset			Wilt dataset		
	accuracy	AUC	F1 Score	accuracy	AUC	F1 Score	accuracy	AUC	F1 Score	accuracy	AUC	F1 Score
DPGAN	F	F	F	14.7186%	0.1309	0.1861	39.7299%	0.3278	0.4616	30.5785%	0.2904	0.3958
Pate-GAN	F	F	F	23.3766%	0.2348	0.2927	26.8186%	0.2811	0.3987	31.8182%	0.3746	0.4085
CTGAN	3.5249%	0.0362	0.0632	23.7373%	0.3389	0.2652	20.6751%	0.1396	0.2005	23.9096%	0.1388	0.2461
CTAB-GAN	**2.6432%**	**0.0219**	**0.0244**	10.4618%	0.1082	0.1015	**14.2560%**	**0.1087**	**0.1512**	4.9128%	0.1754	0.2412
CTAB-GAN+	5.5784%	0.0459	0.0530	7.5758%	0.0487	0.0601	19.4543%	**0.1216**	0.2019	3.3747%	0.1725	0.2411
CG-TGAN	4.9137%	0.0693	0.0748	**-0.5772%**	**-0.0117**	**-0.0131**	**15.7581%**	0.1424	**0.1703**	**0.7805%**	**0.0662**	**0.0583**

Table 2: Machine Learning Utility Results on Classification Datasets The smaller the number in the table means the better performance. ”F” means that the quality of the synthesized data is too low to evaluate.

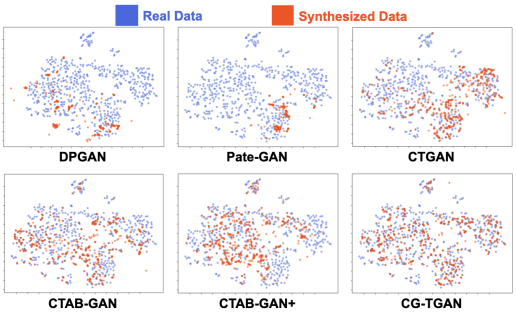

Figure 3: t-SNE projection on Diabetes dataset.

	Abalone	Insurance	King	Adult	Diabetes	Gesture	Wilt
CTGAN	0.6741	0.5787	0.7696	0.6240	0.6978	0.8746	0.7556
CTAB-GAN+	0.7102	0.5451	0.8821	0.6092	0.7493	0.8680	0.6915
CoDi	0.6178	0.6012	0.8029	0.6815	0.6526	0.8054	0.6354
TabDDPM	0.6148	0.3821	0.7523	0.4843	0.6454	0.7238	0.5869
CG-TGAN	0.6428	0.3839	0.7458	0.6271	0.6352	0.8843	0.5485

Table 3: NNDR. Comparison of the privacy protection ability of each tabular data synthesizer model.

Privacy Protection Table 3 shows the privacy protection evaluation results. DPGAN and Pate-GAN have been excluded due to poor quality results. CTAB-GAN has been excluded due to its inability to generate regression datasets.

The privacy protection of synthesizers must be evaluated

		Abalone	Insurance	King		Adult	Diabetes	Gesture	Wilt
CG-TGAN	MAPE	-0.0046	**0.0489**	**-0.0006**	acc	4.9137%	**-0.5772%**	15.7581%	0.7805%
	MAE	-0.1174	**1.115e+2**	**-9.325e+3**	AUC	0.0693	**-0.0117**	0.1424	0.0662
	R2	0.1274	0.0065	**0.1281**	F1	0.0748	**-0.0131**	0.1703	0.0583
CoDi	MAPE	-0.0412	-0.2698	-0.2459	acc	7.2142%	4.7619%	14.6610%	0.9297%
	MAE	-0.2736	-7.691e+2	-8.740e+5	AUC	0.0484	0.0158	0.0835	**-0.0015**
	R2	0.1281	0.0449	0.3473	F1	0.0623	0.0366	0.1312	0.0189
TabDDPM	MAPE	**-0.0027**	-0.0268	-0.1505	acc	**0.5780%**	2.2366%	**4.7763%**	**-0.0688%**
	MAE	**-0.0246**	-1.484e+2	-6.236e+5	AUC	**0.0097**	0.0081	**0.0327**	0.0060
	R2	**0.0170**	**-0.0015**	0.2726	F1	**0.0087**	0.0246	**0.0439**	**-0.0002**

Table 4: Comparison between CG-TGAN and diffusion-based models in terms of machine learning utilities.

relatively. We found that as the quality of the synthesized data increases, the NNDR value tends to decrease. Because low-quality data is free from privacy leakage, but data with similar quality to real data is at risk of privacy leakage. To evaluate CG-TGAN's privacy protection ability, comparing it with comparable performance synthesizers is necessary. So, we perform a more detailed comparison with TabDDPM, a synthesizer with comparable performance, in the next section. It will provide a comprehensive understanding of CG-TGAN's privacy protection ability.

When comparing NNDR values with GAN-based models, CG-TGAN's NNDR values are comparable on the Abalone, King, Adult, Diabetes, and Gesture datasets. We can see that CG-TGAN has a higher NNDR value in the Gesture dataset, despite having better machine learning utility performance.

Comparison with Diffusion-based models

Machine Learning Utility Table 4 shows the results of machine learning utility. CG-TGAN outperforms CoDi, and it is comparable to TabDDPM. The dominance of CG-TGAN in three of seven experimental datasets (Insurance, King, and Diabetes) confirms that CG-TGAN is comparable to TabDDPM in terms of machine learning utility. The dominance in the Insurance and Diabetes datasets means CG-TGAN can generate better synthetic data than TabDDPM for small tabular datasets with 1000 or fewer rows.

Privacy Protection TabDDPM has a worse privacy protection performance than CG-TGAN. Table 3 shows CG-TGAN has higher NNDR values than TabDDPM for 4 out of 7 datasets. To provide a more comprehensive understanding of the privacy protection performance difference between CG-TGAN and TabDDPM, we utilized the distance to the closest record(DCR)(Zhao et al. 2021) metric. DCR is the Euclidean distance between the synthetic data and the closest real data. A higher DCR value can be explained as being safe from privacy breaches. In 6 out of 7 datasets, CG-TGAN has higher DCR values. At Insurance and King datasets, CG-TGAN has higher DCR values, even though its machine learning utility is better than TabDDPM. This means that CG-TGAN is safer than TabDDPM.

	Abalone	Insurance	King	Adult	Diabetes	Gesture	Wilt
TabDDPM	0.1569	0.1542	0.6322	0.1547	**0.9756**	0.2183	0.1339
CG-TGAN	**0.3418**	**0.3181**	**0.8419**	**0.5520**	0.8636	**0.6530**	**0.2375**

Table 5: DCR comparison between CG-TGAN and TabDDPM.

Ablation Analysis

	Abalone	Insurance	King	Adult	Diabetes	Gesture	Wilt
GC1	0.2001	0.5080	0.3117	8.9320%	7.6479%	15.6455%	3.7764%
GC2	0.1405	0.0208	0.1670	8.1301%	2.3809%	15.6849%	1.8709%
GC3	**0.1274**	**0.0065**	**0.1281**	6.7256%	**-0.5772%**	**15.7581%**	1.7791%
GC4	0.1605	0.0272	0.1357	**4.9137%**	2.5252%	F	**0.7805%**
w/o resi	0.1505	0.0354	0.1387	7.4169%	2.6695%	16.2419%	1.5036%
w/o cond	0.1362	0.0137	0.1625	5.3866	6.7821%	16.9620%	3.3402%
w rand adj	0.1444	0.2566	0.3328	6.7058	3.6797%	16.0731%	2.8237%

Table 6: Ablation Analysis.

We involved a comprehensive exploration of the appropriate number for CG-TGANs, achieved by varying the number of graph convolutional layers. This allowed us to gain valuable insights into the model's performance under different configurations. The results in Table 6 show that the model generally performed well when using three or four graph convolutional layers. For the Gesture dataset, the difference in performance between one, two, and three graph convolutional layers was insignificant, so we used three graph convolution layers, which generally performed well in other datasets. Also, Table 6 shows that using residual connection improves the performance of CG-TGAN.

We conduct ablation analysis to check the effectiveness of the conditional node and the effectiveness of initializing the parameter matrix(A_P) as $J_N - I_N$. Table 6 shows using conditional nodes and initializing A_P as $J_N - I_N$ contribute to the performance of CG-TGAN. Although it cannot be shown in the table, we confirmed that initializing A_P as $J_N - I_N$ helps the model to converge more stably.

Conclusion

We proposed CG-TGAN. The discriminator learns graph-level task to distinguish whether an input graph is real or fake and node-level task to predict the target node value. The generator learns graph-level task to embed a fake graph macroscopically similar to the real graph and node-level task to synthesize a fake graph with the correct relations among other nodes. CG-TGAN outperforms existing GAN-based models constructed using fully connected or convolutional layers. This means that our starting hypothesis, "if we convert tabular data to graph-structured data and construct the tabular data synthesizer model with GNN, we could generate better synthetic data" is valid. Also CG-TGAN shows comparable machine learning utility performance with TabDDPM. Moreover, CG-TGAN outperforms TabDDPM in terms of privacy protection.

References

Bishop, C. 2006. Pattern recognition and machine learning. *Springer google schola*, 2: 35–42.

Breiman, L. 2001. Random forests. *Machine learning*, 45: 5–32.

Choi, E.; Biswal, S.; Malin, B.; Duke, J.; Stewart, W. F.; and Sun, J. 2017. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, 286–305. PMLR.

Choi, M. 2018. Medical Cost Personal Datasets — kaggle.com. https://www.kaggle.com/datasets/mirichoi0218/insurance. [Accessed 15-05-2024].

De Myttenaere, A.; Golden, B.; Le Grand, B.; and Rossi, F. 2016. Mean absolute percentage error for regression models. *Neurocomputing*, 192: 38–48.

Dhariwal, P.; and Nichol, A. 2021. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34: 8780–8794.

Engelmann, J.; and Lessmann, S. 2021. Conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning. *Expert Systems with Applications*, 174: 114582.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

HARLFOXEM. 2017. Predict house price using regression. https://www.kaggle.com/datasets/harlfoxem/housesalesprediction.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Hoerl, A. E.; and Kennard, R. W. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1): 55–67.

Hwang, E.; Thost, V.; Dasgupta, S. S.; and Ma, T. 2022. An analysis of virtual nodes in graph neural networks for link prediction. In *The First Learning on Graphs Conference*.

Johnson, B. 2014. Wilt. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5KS4M.

Jordon, J.; Yoon, J.; and Van Der Schaar, M. 2018. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

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

Kotelnikov, A.; Baranchuk, D.; Rubachev, I.; and Babenko, A. 2023. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, 17564–17579. PMLR.

Lee, C.; Kim, J.; and Park, N. 2023. Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis. In *International Conference on Machine Learning*, 18940–18956. PMLR.

Li, Z.; Cui, Z.; Wu, S.; Zhang, X.; and Wang, L. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM international conference on information and knowledge management*, 539–548.

Luo, Y.; Zhou, H.; Tu, W.-W.; Chen, Y.; Dai, W.; and Yang, Q. 2020. Network on network for tabular data classification in real-world applications. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2317–2326.

Madeo, R. C.; Lima, C. A.; and Peres, S. M. 2013. Gesture unit segmentation using support vector machines: segmenting gestures from rest positions. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 46–52.

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mottini, A.; Lheritier, A.; and Acuna-Agost, R. 2018. Airline passenger name record generation using generative adversarial networks. *arXiv preprint arXiv:1807.06657*.

Nash, S. T. T. S. C. A., Warwick; and Ford, W. 1995. Abalone. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55C7W.

Odena, A.; Olah, C.; and Shlens, J. 2017. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, 2642–2651. PMLR.

Park, N.; Mohammadi, M.; Gorde, K.; Jajodia, S.; Park, H.; and Kim, Y. 2018. Data synthesis based on generative adversarial networks. *arXiv preprint arXiv:1806.03384*.

Pearl, J. 2009. *Causality*. Cambridge university press.

Quinlan, J. R. 1986. Induction of decision trees. *Machine learning*, 1: 81–106.

Sigillito, V. 2014. OpenML — openml.org. https://www.openml.org/search?type=data&sort=runs&id=37&status=active. [Accessed 15-05-2024].

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1): 267–288.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y.; et al. 2017. Graph attention networks. *stat*, 1050(20): 10–48550.

Xie, L.; Lin, K.; Wang, S.; Wang, F.; and Zhou, J. 2018. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*.

Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; and Veeramachaneni, K. 2019. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32.

Zhao, Z.; Kunar, A.; Birke, R.; and Chen, L. Y. 2021. Ctab-gan: Effective table data synthesizing. In *Asian Conference on Machine Learning*, 97–112. PMLR.

Zhao, Z.; Kunar, A.; Birke, R.; and Chen, L. Y. 2022. Ctab-gan+: Enhancing tabular data synthesis. *arXiv preprint arXiv:2204.00401*.