A Data Augmentation Model

The data augmentation model conditionally generates augmented data, i.e. $\hat{x} = F_\theta(z, X_i = x_i; G_\phi)$. To allow conditioning on any variable during the generation stage, we assume a directed acyclic graph (DAG). This means that variables are generated sequentially, following topological order π_{G_ϕ}, which can be described through a SEM (Pearl, 2009):

$$\hat{x}_i = f_i(\hat{\mathcal{N}}(i), z_i)$$

In contrast to the synthetic data model, the generation of each variable is conditioned on the predicted value of its parents. [4] Each variable is transformed through a specific embedding function $h_i^{(0)} = f_{\theta_i}(\hat{x}_i) \; \forall \, i \in [d]$, where $h_i^{(0)} \in \mathbb{R}^{d_0}$ and we implement f_{θ_i} using a one layer MLP. This is followed by one round of message passing, where the embedding update function is implemented using a single-layer MLP. We aggregate parent embeddings using the mean, equivalent to taking a weighted average:

$$\begin{aligned} m_j^{(1)} &= \sigma_\theta^{(1)}(\cdot) = h_j^{(0)} \\ h_{\mathcal{N}(i)}^{(1)} &= \oplus^{(1)}(\cdot) = \text{mean}\left(\{G_{j,i} m_j^{(1)}, \; \forall \, j \in \mathcal{N}(i)\}\right) \\ \hat{x}_i &= \gamma_\theta^{(1)}(\cdot) = \tanh\left(W_u^{(1)} \times h_{\mathcal{N}(i)}^{(1)}\right) \end{aligned}$$

The output after message passing is taken to be the prediction of the current variable $\hat{x}_i$. This model is trained using the GAN adversarial loss (eq. (5)). Additionally, we use the continuous DAG penalty introduced in (Zheng et al., 2018) to force a DAG to be recovered:

$$\max_{F_\theta} \min_{D} \underbrace{\mathbb{E}[\log D(F_\theta(Z; G_\phi)) + \log(1 - D(X))]}_{GAN Loss} + \underbrace{\lambda(\text{tr}(\exp(G_\phi \circ G_\phi)) - d)}_{Regularisation} \tag{5}$$

To generate augmented data, we randomly sample a subset of the variables, and for each of which, we sample a conditioning value *uniformly* from the support of that variable, $a \sim \mathcal{U}\left(\min(X_i), \max(X_i)\right)$. We then generate augmented data by conditioning chosen variables on sampled values, $\hat{x} \sim p(X|X_i = a) = F_\theta(z, X_i = a; G_\phi)$ (Pearl, 2009). We note that this conditional generation corresponds to sampling *out-of-distribution*, as we are sampling from a different distribution that is defined on the same support as the training.

B Implementation Details

B.1 Models and Evaluation

Models. All models are implemented in PyTorch (Paszke et al., 2017). The likelihood term is computed using Cross Entropy or MSE Loss for categorical and continuous variables, respectively. We use $d_0 = 3$ for all experiments and $d_1 \in \{32, 64, 128\}$ depending on the complexity of the data. Hyperparameters include batch size $\in \{64, 128, 256\}$, learning rate $\in \{1\text{e}{-3}, 1\text{e}{-2}\}$, and are chosen through cross-validation. The data is split 60-20-20 into train, validation and test sets and reported results are averaged over 10 runs. Training is performed using the Adam (Kingma & Ba, 2014). All experiments are run on an NVIDIA Tesla K40C GPU, taking less than an hour to complete.

For the synthetic data model, we consider regularization penalty $\lambda \in \{1\text{e}{-3}, 1\text{e}{-2}, 1\text{e}{-1}\}$. For data augmentation, we gradually increase the value of λ as optimization proceeds, so that non-DAGs are heavily penalized (similar to (Zheng et al., 2018)). A specific schedule is employed to increase λ: in each stage of the schedule, the objective is optimized with fixed values of λ for t steps, before λ is updated, $\lambda' \leftarrow \alpha * \lambda$, and the process repeats. We consider starting values $\lambda_0 \in \{10, 100, 1000\}$, $\alpha = 10$ and $t = 1000$. The topological ordering of the nodes is obtained by running a topological sort on the adjacency matrix. All models are trained for a maximum of 5000 epochs, with early stopping if no improvements on the validation set for 100 epochs.

Evaluation. We partition the observed dataset $\mathcal{D}$ into a training set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$, and train generative models on $\mathcal{D}_{train}$. Using the trained models, we generate a synthetic data set $\mathcal{D}_{syn}$,

[4] $\mathcal{N}(i) = Pa(i)$, the neighborhood is strictly the set of parent variables due to the DAG.

which has the same number of samples as $\mathcal{D}_{test}$. We then evaluate the aforementioned desiderata on $(\mathcal{D}_{test}, \mathcal{D}_{syn})$. To evaluate **diversity**, we employ the three-dimensional metric, α-precision, β-recall, and authenticity, which assesses whether the samples are realistic, diverse enough to cover the variability in real data, and generalization performance respectively (Alaa et al., 2021). Each metric evaluates a different aspect of synthetic data diversity, and we average the three metrics to obtain a holistic score. **Fidelity** is evaluated by training post-hoc classifiers to distinguish samples from the original and generated datasets. Specifically, we train a two-layer MLP, XGB classifier and GMM classifier and average the classification AUROC. To evaluate **utility**, we report average performance achieved by three downstream prediction models (linear model, two-layer MLP, and XGB model) train on the $\mathcal{D}_{syn}$ and evaluated on $\mathcal{D}_{test}$. We report the change in AUROC of models trained on $\mathcal{D}_{syn}$ and those trained on $\mathcal{D}_{train}$.

B.2 Synthetic Data Benchmarks

In this subsection, we provide further details on the benchmarks we compare against: including Bayesian networks (`BN`) (Pearl, 2011), GAN-based (Xu et al., 2019), `TableGAN` (Park et al., 2018) and VAE-based (Xu et al., 2019; Patki et al., 2016) and a normalizing flow `NFLOW` (Rezende & Mohamed, 2015). Additionally, we also consider data augmentation methods: Gaussian noise `InNoise` (Krizhevsky et al., 2012); `MixUp` (Zhang et al., 2018) and `SwapNoise` (Jahrer, 2019).

`BN` (Pearl, 2011). We train `BN` in two stages, where the first stage learns the network structure and the second stage performs learning based on the returned DAG. For structure learning, we consider constraint-based PC algorithm (Spirtes & Glymour, 1991), Hill-Climb algorithm (Heckerman et al., 1995) and Max-Min Hill-Climb (Tsamardinos et al., 2006). Once a DAG is returned, the conditional probabilities are learned through maximum likelihood estimation, where continuous variables are assumed to come from a linear Gaussian conditional probability distribution (CPD) and discrete variables from a discrete CPD.

`CTGAN` (Xu et al., 2019), `TableGAN` (Park et al., 2018). For `CTGAN`, we use an MLP with two ReLU-activated hidden layers to implement the generator. The number of units in each layer, $\in \{128, 256\}$ depending on the complexity of the data. Similarly, we employ an MLP with two ReLU-activated hidden layers to implement the discriminator, where the number of hidden units $\in \{128, 256\}$. The hyperparameters are tuned according to the recommended settings in (Patki et al., 2016). `TableGAN` is implemented using a Deep Convolution GAN with recommended settings in (Park et al., 2018), where the generator has three deconvolutional layers, and the discriminator has three convolutional layers.

`TVAE` (Patki et al., 2016). The VAE-based model is implemented with an encoder with two ReLU-activated layers, where the number of units in each layer of the encoder $\in \{128, 256\}$, depending on complexity. The decoder similarly has two hidden layers with dimensionality $\in \{128, 256\}$. We use a d_h dimensional latent space that is normally distributed and a standard normal prior, where $d_h \in \{32, 64\}$.

`NFLOW` (Rezende & Mohamed, 2015). We implement the normalizing flows using the rational-quadratic transform introduced in (Durkan et al., 2019). Specifically, it is implemented using an MLP with 2, 128-dimensional hidden layers and permutation operations. A standard normal base distribution is employed, and the flow is run with 500 steps.

B.3 Data Augmentation Benchmarks

`InNoise` (Krizhevsky et al., 2012), `MixUp` (Zhang et al., 2018), `SwapNoise` (Jahrer, 2019). For `InNoise`, we add zero-centered Gaussian noise $\varepsilon \sim N(0, \sigma^2)$ to the inputs, where we consider $\sigma \in \{0.01, 0.1, 1\}$. For `SwapNoise`, we randomly swap 10% of elements between two inputs. `MixUp` is implemented by randomly combining two samples $\hat{x} = \lambda x' + (1 - \lambda)x''$, where $x', x'' \sim \mathcal{D}_{train}$ and $\lambda \sim \text{Beta}(0.2, 0.2)$.

DAFS (DeVries & Taylor, 2017). We train an autoencoder, where the encoder and decoder are both implemented as MLPs with two ReLU-activated hidden layers. We take the feature vector at the output of the encoder c_i and randomly apply one of three possible operations (1) add Gaussian noise $\varepsilon \sim (0, \sigma^2)$, (2) interpolation $\lambda(c_j - c_i) + c_i$, or (3) extrapolation $\lambda(c_i - c_j) + c_i$ where $\lambda = 0.5$ as

Table 4: **Experimental datasets.** Description of experimental datasets.

Dataset	Description	Number of instances	Number of features
Adult (Kohavi et al., 1996)	Census data	48842	15
Breast (Street et al., 1993)	Breast cancer	569	32
Covertype (Blackard & Dean, 1999)	Forest cover	581012	54
Credit (Hofmann, 1994)	Credit risk	1000	20
ECOLI (Schäfer & Strimmer, 2005)	Functional genomics	2000	46
MAGIC-IRRI (Scutari et al., 2014)	Plant genetics	2000	64
Red (Cortez et al., 2009)	Wine quality	1599	12
White (Cortez et al., 2009)	Wine quality	4898	12
Mice (Higuera et al., 2015)	Protein expression	1080	82
Musk (Dua & Graff, 2017)	Musk molecules	6598	168

suggested by the authors. Augmented samples x' are then obtained by passing the altered feature vector through the decoder.

B.4 Datasets

We use 10 datasets in total, including 8 UCI datasets (Dua & Graff, 2017),[5] specifically `Adult`, `Breast`, `Covertype`, `Credit`, `White`, `Red`, `Mice`, `Musk` and 2 Bayesian Network repository datasets (Koller & Friedman, 2009),[6] specifically `ECOLI` and `MAGIC-IRRI`. A summary of the datasets, including dataset description, the dimensionality, and number of samples, is presented in Table 4.

B.5 Prior Knowledge

The use of an explicit graph to guide generation allows for a variety of prior knowledge to be incorporated through the adjacency matrix. Here, we describe a few options of incorporating domain expertise:

- **Sparsity.** The learned graph can be sparse such that variables only depend on a small subset of other variables. Mathematically, $\mathcal{R}(G_\phi) = ||G_\phi||_p$, where $||\cdot||_p$ denotes the L_p matrix norm.
- **Dependence.** Partial knowledge about the dependencies between features can be encoded through a graph prior G_0, i.e., $\mathcal{R}(G_\phi) = ||G_\phi - G_0||_p$.
- **Graph types.** Graphs of specific types can be learned. For example, if an undirected graph is assumed, we can employ a symmetric prior $||G_\phi - G_\phi^T||_p$. Alternatively, we can use the DAG penalty (Zheng et al., 2018) to encourage learning a directed, acyclic graph (DAG), $\mathcal{R}(G_\phi) = \text{tr}(\exp(G_\phi \circ G_\phi) - D)$, where $tr(\cdot)$ is the matrix trace and D is the number of variables.
- **Connectivity.** Encourage different patterns of connectivity through penalty on degree of each variable $||D_\phi - D_0||_p$, where $D \in \mathbb{R}^d$ is the degree of each variable.

C Additional Experiments

In this section, we provide additional results to comprehensively evaluate our proposed methods, specifically:

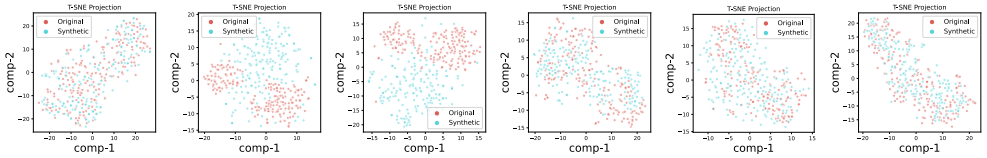
1. **Additional datasets**: §C.1 evaluates synthetic data performance on 4 additional datasets.
2. **Visualizations**:§C.2 visualizes t-SNE projections on original and synthetic datasets to qualitatively investigate quality and examines learned adjacency matrices.
3. **Sensitivity**: §C.3 investigates performance sensitivities according to data size and feature counts.
4. **Data augmentation**: §C.4 describes the best model performance after data augmentation.

.

[5] https://archive.ics.uci.edu/ml/datasets.php
[6] https://www.bnlearn.com/bnrepository/

Table 5: **Diversity, fidelity, and utility of synthetic data.** Bold indicates the best performance.

	Dataset	ECOLI	MAGIC-IRRI	Red	White	Mice	Musk
Diversity (the higher the better)	BN	$\mathbf{0.57 \pm 0.06}$	$\mathbf{0.67 \pm 0.04}$	0.64 ± 0.05	0.63 ± 0.07	$\mathbf{0.63 \pm 0.02}$	0.60 ± 0.06
	CTGAN	0.38 ± 0.06	0.33 ± 0.10	0.45 ± 0.08	0.51 ± 0.03	0.42 ± 0.02	0.51 ± 0.05
	TableGAN	0.38 ± 0.08	0.33 ± 0.07	0.46 ± 0.08	0.53 ± 0.02	0.41 ± 0.04	0.56 ± 0.03
	TVAE	0.45 ± 0.09	0.61 ± 0.02	0.59 ± 0.05	0.61 ± 0.03	0.57 ± 0.03	0.58 ± 0.05
	NFLOW	0.56 ± 0.08	0.62 ± 0.07	0.59 ± 0.04	0.63 ± 0.03	0.54 ± 0.06	0.62 ± 0.08
	GOGGLE	$\mathbf{0.57 \pm 0.05}$	0.63 ± 0.09	$\mathbf{0.69 \pm 0.07}$	$\mathbf{0.67 \pm 0.05}$	$\mathbf{0.63 \pm 0.04}$	$\mathbf{0.65 \pm 0.04}$
Fidelity (the lower the better)	BN	$\mathbf{0.39 \pm 0.07}$	$\mathbf{0.40 \pm 0.03}$	0.56 ± 0.07	$\mathbf{0.58 \pm 0.01}$	0.61 ± 0.04	0.83 ± 0.03
	CTGAN	0.74 ± 0.09	0.73 ± 0.07	0.77 ± 0.09	0.79 ± 0.04	0.72 ± 0.02	0.78 ± 0.06
	TableGAN	0.74 ± 0.02	0.73 ± 0.06	0.77 ± 0.05	0.82 ± 0.07	0.85 ± 0.02	0.77 ± 0.05
	TVAE	0.74 ± 0.02	0.69 ± 0.05	0.67 ± 0.07	0.63 ± 0.05	0.73 ± 0.01	0.79 ± 0.02
	NFLOW	0.70 ± 0.03	0.70 ± 0.08	0.67 ± 0.04	0.73 ± 0.05	0.77 ± 0.05	0.80 ± 0.03
	GOGGLE	0.60 ± 0.03	0.69 ± 0.09	$\mathbf{0.55 \pm 0.03}$	0.65 ± 0.06	$\mathbf{0.60 \pm 0.03}$	$\mathbf{0.73 \pm 0.05}$
Utility (the higher the better)	BN	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.05 \pm 0.00}$	0.00 ± 0.00	-0.15 ± 0.05	$\mathbf{0.00 \pm 0.00}$	-0.18 ± 0.04
	CTGAN	-0.20 ± 0.03	-0.13 ± 0.01	$\mathbf{0.02 \pm 0.00}$	-0.08 ± 0.00	-0.10 ± 0.02	-0.13 ± 0.05
	TableGAN	-0.18 ± 0.06	-0.10 ± 0.05	$\mathbf{0.02 \pm 0.00}$	-0.14 ± 0.02	-0.19 ± 0.03	-0.13 ± 0.04
	TVAE	-0.06 ± 0.01	0.00 ± 0.00	-0.07 ± 0.01	-0.02 ± 0.00	-0.07 ± 0.03	$\mathbf{-0.06 \pm 0.02}$
	NFLOW	-0.05 ± 0.01	-0.02 ± 0.00	0.01 ± 0.00	-0.15 ± 0.05	-0.04 ± 0.01	-0.16 ± 0.04
	GOGGLE	-0.02 ± 0.00	0.01 ± 0.00	$\mathbf{0.02 \pm 0.00}$	$\mathbf{-0.01 \pm 0.00}$	$\mathbf{0.00 \pm 0.01}$	-0.12 ± 0.02



(a) BN (b) CTGAN (c) TableGAN (d) TVAE (e) NFLOW (f) GOGGLE

Figure 4: **t-SNE projection on Breast dataset.**

C.1 Additional Results

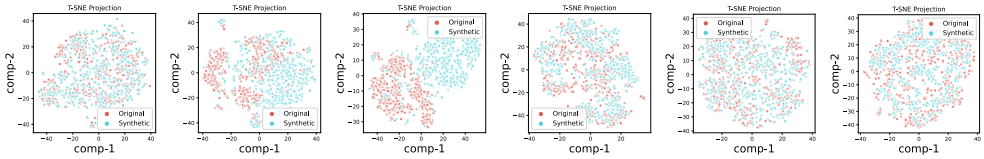
We assess the quality of synthetic dataset using the same desiderata introduced in §5.1, namely *diversity*, *fidelity*, and *utility*. We use six additional datasets, `ECOLI`, `MAGIC-IRRI`, `Red`, `White`, `Mice`, and `Musk`. The results are reported in Table 5.

We note that `BN` achieves the best performance on `ECOLI` and `MAGIC-IRRI`, which is reasonable as those datasets are generated according to a known Bayesian network, and `BN` models have a natural advantage. On those two datasets, `GOGGLE` is able to consistently outperform other deep generative models. On `Red` and `White`, `GOGGLE` achieves superior performance against other benchmarks. On the contrary, `BN`, our closest competitor, achieve worse performance as the underlying DAG assumptions become too restrictive. Additionally, we highlight that models trained on synthetic data generated by `GOGGLE` consistently achieves similar performance to those trained on real datasets, indicating strong data utility

C.2 Visualization of Synthetic Data Results

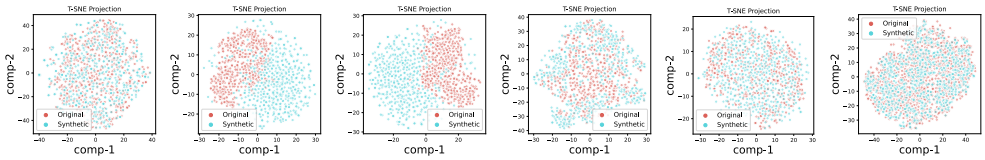
In Figures 4 to 7, we observe that synthetic data generated by `GOGGLE` exhibit markedly better overlap with the original dataset than other benchmarks using t-SNE for visualization. We note that the GAN-based models, specifically `CTGAN` and `TableGAN` exhibit mode collapse behaviour and the `TVAE` and `NFLOW` can fail to match the underlying distribution (on `ECOLI` and `Breast`, respectively). Additionally, we plot the adjacency matrix of trained models in Figure 8, where a sparsity regularization term was applied to all models to encourage sparsely connected graphs.

We visualize the learned graphs on `Credit` and `Breast` in Figure 9. The `Breast` dataset (Dua & Graff, 2017) contains numeric features extracted from images of a breast mass. We note that the *target* variable (diagnosis of tumor) has a high degree of connectivity, and dependent on various physical properties of the tumor, including *mean perimeter* and *mean compactness*. We similarly observe informative variables identified in the `Credit` dataset (Dua & Graff, 2017) , where the *account balance* depends on *occupation*, *credit amount*, and *length of current employment*.



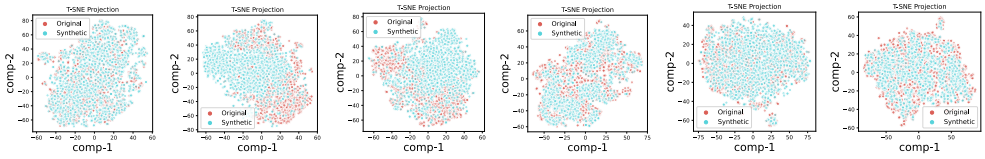
(a) BN (b) CTGAN (c) TableGAN (d) TVAE (e) NFLOW (f) GOGGLE

Figure 5: **t-SNE projection on Red dataset.**



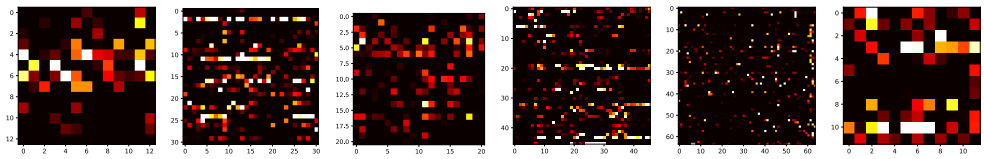
(a) BN (b) CTGAN (c) TableGAN (d) TVAE (e) NFLOW (f) GOGGLE

Figure 6: **t-SNE projection on ECOLI dataset.**



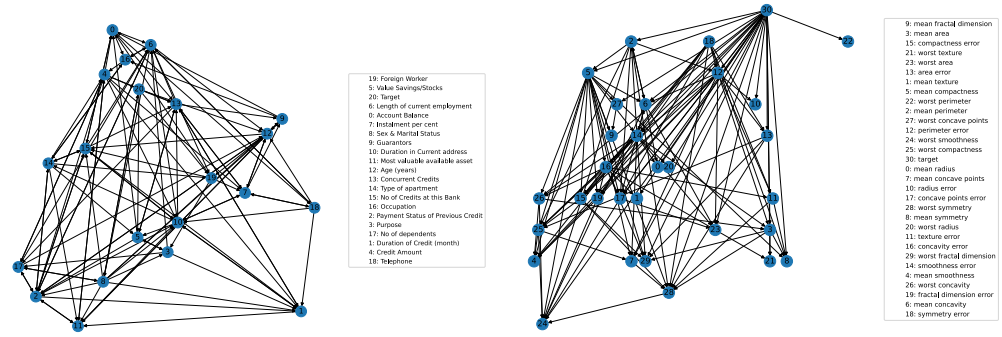
(a) BN (b) CTGAN (c) TableGAN (d) TVAE (e) NFLOW (f) GOGGLE

Figure 7: **t-SNE projection on White dataset.**



(a) Adults (b) Breast (c) Credit (d) ECOLI (e) MAGIC (f) White

Figure 8: **Learned adjacency matrices.**



(a) Credit (b) Breast

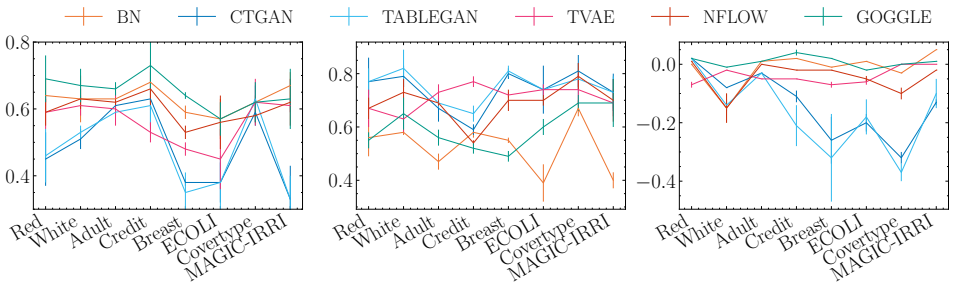
Figure 9: **Learned graphs.**

C.3 Sensitivity Analysis

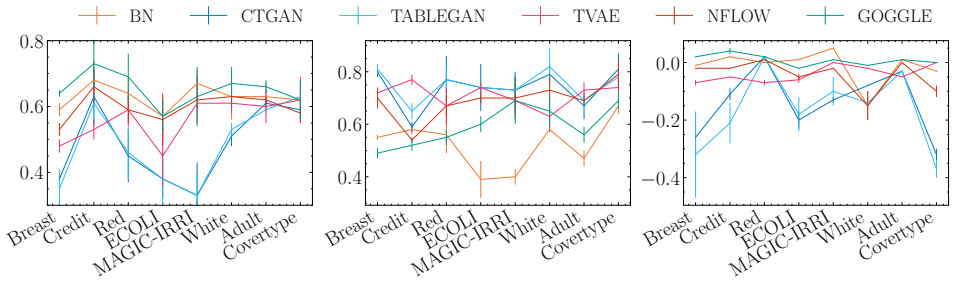
Lastly, we are interested in identifying settings where GOGGLE excel as a generative model. Specifically, we are interested in understanding sensitivities of model performance with respect to the effects of feature counts and number of samples in the dataset. We compare our model against the

Table 6: **Data augmentation**. AUROC on $\mathcal{D}_{test}$ of models trained on augmented data. Bold indicates the best performance.

Dataset	Adult	Breast	Covertype	Credit
Baseline	0.70 ± 0.09	0.96 ± 0.01	0.59 ± 0.13	0.65 ± 0.02
InNoise	0.68 ± 0.05	0.97 ± 0.01	0.61 ± 0.10	0.64 ± 0.03
MixUp	0.66 ± 0.08	0.90 ± 0.01	0.56 ± 0.09	0.65 ± 0.02
SwapNoise	0.65 ± 0.05	0.91 ± 0.01	0.58 ± 0.08	0.63 ± 0.01
FSAug	0.71 ± 0.06	0.96 ± 0.01	0.60 ± 0.13	$\mathbf{0.67 \pm 0.02}$
GOGGLE-SD	0.70 ± 0.05	0.97 ± 0.01	0.60 ± 0.10	0.66 ± 0.01
GOGGLE	$\mathbf{0.72 \pm 0.03}$	$\mathbf{0.98 \pm 0.00}$	$\mathbf{0.62 \pm 0.10}$	$\mathbf{0.67 \pm 0.02}$



(a) Feature Count



(b) Number of Samples

Figure 10: **Sensitivity analysis.** Evaluating synthetic data based on **(left) diversity** (the higher, the better), **(middle) fidelity** (the lower, the better), and **(right) utility** (the higher, the better). Datasets are sorted according to **(a)** increasing feature counts, and **(b)** increasing number of samples.

benchmarks in Figure 10. Here, the datasets are shown on the x-axis and are sorted in order of increasing feature count, and increasing number of samples (see Table 4 for more on datasets). We note that the advantage of GOGGLE is more noticeable when there are less number of samples (i.e., on Breast, Credit and Red). In the regime with larger number of samples, all models exhibit similar performance, although GOGGLE still achieves performance improvements. Furthermore, models achieve similar performance when the number of features is low. However, when the number of features increases, the performance of GAN-based models deteriorate. This is interesting, and a potential logical explanation is that they are overfitting to the training data.

C.4 Evaluation of Augmented Data

We compare popular tabular data augmentation methods by inspecting downstream model performance. Specifically, we generate augmented data $\mathcal{D}_{aug}$ from $\mathcal{D}_{train}$ (and has the same number of samples), train predictive models on the combined $\mathcal{D}_{comb} = \{\mathcal{D}_{train}, \mathcal{D}_{aug}\}$, and evaluate performance on $\mathcal{D}_{test}$. We train four downstream prediction models, including linear model, two-layer MLP, RF classifier, and XGB model and report the averaged performance achieved by the four models. We perform data augmentation on GOGGLE by randomly selecting variables to condition on and sampling uniformly from the marginal support of the variable. In Table 6, we observe that augmented data generated by GOGGLE leads to improved generalization performance across all datasets.

D CONNECTION TO PROBABILISTIC STRUCTURE DISCOVERY

There are several parallels between our work and *probabilistic graph discovery*, which aims to recover the true probabilistic graphical model (PGM) underlying observed data (Zhou, 2011; Drton & Maathuis, 2017). We propose a generative model for tabular data that learns and leverages an underlying graph to improve the performance of data synthesis. Importantly, our model does not recover the true PGM from data if the data is indeed generated by a PGM (i.e. it does not perform PGM structural discovery). Additionally, the message passing computation is not an instance of or an approximation to a probabilistic inference routine. Specifically, we take advantage of the sparse and compact representations of graphical models to learn better generative models, incorporate prior knowledge, and perform conditional generation. We do so by incorporating a graph as explicit structure into the generative process. We summarize the key distinctions between our work and the probabilistic structure learning literature.

The graph learned in GOGGLE encodes conditional dependence structure between variables (global Markov property), in the same sense that PGMs reflect allowed conditional dependencies. The key distinction between our approach and probabilistic structural learners is that we only require an approximate structure. In contrast, structural learners aim to recover unique graphs that are close to the true DGP. This learned graph is used to answer probabilistic inference queries, e.g. $P(Y|X = x)$, which requires the graph to be correct. In order to recover a unique graph, structural learns generally assume certain graph, or distributions, or relationships between variables.

Our objective is to learn an approximate graph that models associational dependence and can guide generation. Therefore, we do not need to make similar assumptions that unnecessarily restrict the class of learnable distributions and can lead to a miss-specified model. Additionally, we emphasize that our proposed method is not designed to perform sampling-based probabilistic inference. Due to the different objectives, we make minimal assumptions on the *graph type*, *variable distribution*, and *functional relations*.

	Probabilistic Graph Discovery	**GOGGLE**
Commonality	Edges reflect allowed conditional dependencies between variables	
Objective	Recover unique probabilistic graph underlying observed data	Learn approximate graph describing dependencies between variables
Evaluation Metric	Quality of discovered graph: graph distance measure (Peters & Bühlmann, 2015; Shimizu et al., 2011), edge classification metric	Quality of synthetic data: diversity, fidelity, and utility (Alaa et al., 2021)
Application	Probabilistic inference $P(Y\|X = x)$	Generate conditional synthetic data $x \sim P_\theta(X)$
Assumptions	Specific graph types (i.e. directed or undirected)	Arbitrary graph types (i.e. mixed, directed, or undirected)
	Distributional assumptions on variables (e.g. Gaussian)	No assumptions on variable distribution
	Assumptions on functional relationships between variables (e.g. Linear with Gaussian additive noise)	No assumptions on functional relationships model
Representative Works	**UGM**: Chow-Liu algorithm (Chow & Liu, 1968), graphical LASSO (Banerjee et al., 2008), neighborhood selection (Meinshausen & Bühlmann, 2006). **DGM**: score-based (Chickering, 2002), constraint-based (Spirtes et al., 2000), hybrid (Tsamardinos et al., 2006)	**Deep generative**: GAN-based (Xu et al., 2019), VAE-based (Xu & Veeramachaneni, 2018). **Non-neural**: BN, mixture models, copula