A NETWORK ARCHITECTURE OF SGM FOR TABULAR DATA

In this section, we provide the neural network architecture of score-based generative model for tabular synthesis, which we used in our experiment for SGMs. The network $F_{\boldsymbol{\theta}_k}(\mathbf{x}(t), t)$ for each k is as follows:

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{x}(t), \\ \mathbf{h}_i &= \omega(\texttt{H}_i(\mathbf{h}_{i-1}, t) \oplus \mathbf{h}_{i-1}), 1 \leq i \leq d_L \\ F_{\boldsymbol{\theta}_k}(\mathbf{x}(t), t) &= \texttt{FC}(\mathbf{h}_{d_L}), \end{aligned}$$

where $\mathbf{x}(t)$ is a record (or a row) at time t in tabular data and ω is an activation function. d_L is the number of hidden layers. The layer $\texttt{H}_i(\mathbf{h}_{i-1}, t)$ is as follows:

$$\texttt{H}_i(\mathbf{h}_{i-1}, t) = \texttt{FC}_i(\mathbf{h}_{i-1}) \odot \psi(\texttt{FC}_i^{gate}(t) + \texttt{FC}_i^{bias}(t)),$$

where $\odot$ means the element-wise multiplication, $\oplus$ means the concatenation operator, ψ is the Sigmoid function, and FC is a fully connected layer.

The model architecture is a straightforward adaptation of the score-based generative model proposed by (Song et al., 2021). Therefore, we largely utilize the framework of (Song et al., 2021), with modifications made to the data input module to suit tabular data. Additionally, for the composition of layers, we strictly adhere to the approach used in CTGAN, a successful method for tabular data synthesis. Since we have borrowed a substantial portion of the implementation from existing methods, we do not claim that designing the architecture of SGM for tabular data is our original contribution. For studying our proposed training framework, we use the above SGM.

B DETAILED EXPERIMENTAL SETTINGS FOR REPRODUCIBILITY

Our software and hardware environments are as follows: UBUNTU 18.04 LTS, PYTHON 3.9.16, PYTORCH 1.10.0, CUDA 11.3, and NVIDIA Driver 470.42.01, i9 CPU, and NVIDIA RTX A5000. Our code for the experiments is mainly based on `https://github.com/yang-song/score_sde_pytorch` (Apache License 2.0).

B.1 DATASETS

We use 10 real-world datasets and 8 baselines for our experiments. The descriptions for the datasets are as follows:

1. `Absenteeism` (Martiniano et al., 2012) predicts the absenteeism time in hours. The dataset consists demographic information.
2. `Bank` (Moro et al., 2014) is to predict whether a client subscribed a term deposit or not. The dataset contains personal financial situations, e.g., whether the person has housing loan or not.
3. `Clave` (Vurkaç, 2011) is to predict the class to which input music belongs. The class is one of the 'Neutral', 'Reverse Clave', 'Forward Clave', and 'Incoherent'.
4. `Concrete` is to calculate strength of cement with 8 variables including characteristics of cement.
5. `Contraceptive` is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics.
6. `Customer` consists of information about the telecommunication company's customers and their groups.
7. `Fish` (Cassotti et al., 2015) aims to predict the acute aquatic toxicity of 908 chemicals towards fathead minnow by using 6 molecular descriptors that characterize the chemical structure of the compounds.
8. `Heart Disease` is to predict the presence of heart disease in a patient, which consists of personal health condition.

Table 5: Dataset information used in our experiments. '#Train' and '#Test' are the numbers of training data and test data, respectively, and 'N_C' and 'N_D' are the numbers of continuous and discrete columns, respectively.

Task	Datasets	**#Train**	**#Test**	N_C	N_D
Binary Classification	Bank	36169	9042	7	10
	Heart Disease	815	203	4	10
	Spambase	3681	920	57	1
Multi Classification	Clave	8639	2159	0	17
	Contraceptive	1179	294	0	10
	Customer	800	200	5	7
	Nursery	10368	2592	0	9
	Obesity	1689	422	8	9
Regression	Absenteeism	592	148	12	9
	Concrete	824	206	8	1
	Fish	727	181	7	0

9. Nursery (Olave et al., 1989) is to rank applications for nursery schools. The dataset contains the family structures, parents' occupation, children's health condition, and so on.
10. Obesity includes data for the estimation of obesity levels in individuals from the countries of Mexico, Peru and Colombia, based on their eating habits and physical condition
11. Spambase (Cranor & LaMacchia, 1998) is a collection of emails labeled as spam or non-spam, with 57 features extracted from the emails to be used in spam filtering.

The statistical information of the datasets is in Table 5. The datasets are available online:

- Absenteeism: https://archive.ics.uci.edu/ml/datasets/Absenteeism+at+work (CC BY 4.0)
- Bank: https://archive.ics.uci.edu/ml/datasets/bank+marketing (CC BY 4.0)
- Clave: https://archive.ics.uci.edu/ml/datasets/Firm-Teacher_Clave-Direction_Classification (CC BY 4.0)
- Concrete: https://www.kaggle.com/datasets/prathamtripathi/regression-with-neural-networking (CC0 1.0)
- Contraceptive: https://archive.ics.uci.edu/dataset/30/contraceptive+method+choice (CC BY 4.0)
- Customer: https://www.kaggle.com/prathamtripathi/customersegmentation (CC0 1.0)
- Fish: https://archive.ics.uci.edu/ml/datasets/QSAR+fish+toxicity (CC BY 4.0)
- Heart Disease: https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset (CC BY 4.0)
- Nursery: https://www.openml.org/search?type=data&sort=runs&id=26&status=active (CC BY 4.0)
- Obesity: https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition (CC BY 4.0)
- Spambase: https://archive.ics.uci.edu/ml/datasets/Spambase (CC BY 4.0)

B.2 BASELINES

The descriptions for the baselines are as follows:

1. MedGAN (Choi et al., 2017) includes non-adversarial losses for discrete medical records.
2. VEEGAN (Srivastava et al., 2017) is a GAN equipped with a reconstructor network, which aims for diverse sampling.
3. CTGAN and TVAE (Xu et al., 2019) handle challenges from the mixed type of variables.
4. TableGAN (Park et al., 2018) is a GAN for tabular data using convolutional neural networks.
5. OCT-GAN (Kim et al., 2021) contains neural networks based on neural ordinary differential equations.
6. RNODE (Finlay et al., 2020) is an advanced flow-based model for image data, and we customize the network to synthesize tabular data.
7. STaSy (Kim et al., 2022a) is a currently proposed score-based generative model for tabular data synthesis.

B.3 DETAILED EVALUATION METHOD

Table 6: Hyperparameters of the base classifiers/regressors

Models	**Hyperparameters**	**Values**
DecisionTree	max_depth	4, 8, 16, 32
	min_samples_split	2, 4, 8
	min_samples_leaf	1, 2, 4, 8
AdaBoost	n_esimators	10, 50, 100, 200
Logistic Regression	solver	lbfgs
	max_iter	10, 50, 100, 200
	C	0.01, 0.1, 1.0
	tol	0.0001, 0.01, 0.1
MLP	hidden_layer_sizes	(100,), (200,), (100, 100)
	max_iter	50, 100
	alpha	0.0001, 0.001
RandomForest	max_depth	8, 16, inf
	min_samples_split	2, 4, 8
	min_samples_leaf	1, 2, 4, 8
XGBoost	n_estimators	10, 50, 100
	min_child_weight	1, 10
	max_depth	5, 10, 20
	gamma	0.0, 1.0

The reported scores for TSTR results in the paper are obtained through the following steps:

1. **Dataset Acquisition**: If the dataset was previously used, the existing train-test split was utilized. Otherwise, a new train-test split is performed, with a ratio of 80% for training data and 20% for testing data.
2. **Fake Data Generation**: We generate the same number of fake records to that of the original training set with the generation methods.
3. **Base Classifier/Regressor Training**: Using the training records generated in Step 2, base classifiers/regressors are trained for prediction tasks. The best hyperparameter settings for each classifier/regressor are determined through a parameter search process. For validation, we use the original training data, i.e., training a classifier/regressor with fake data and predict for training data (Yoon et al., 2019; Alaa et al., 2021; Jarrett et al., 2021; Jeon et al., 2022; Kim et al., 2022a). Table 6 summarizes the considered hyperparameters and

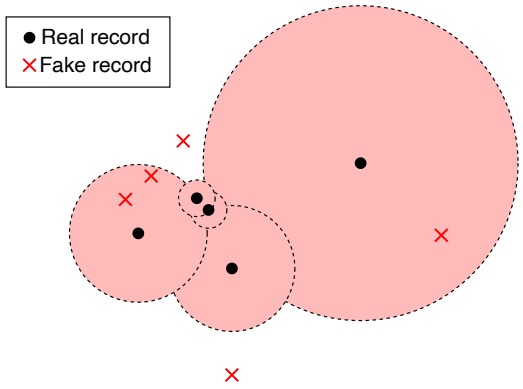

Figure 7: How to calculate coverage. In this example, coverage is $\frac{2}{5}$ since only two real records (out of five) have nearby fake records.

their candidate settings. The classifiers/regressors vary depending on the task: DecisionTree, AdaBoost, Logistic Regression, MLP classifiers, RandomForest, and XGBoost are used for binary classification tasks; DecisionTree, MLP classifiers, RandomForest, and XGBoost for multi-class classification tasks; MLP regressor, RandomForest, and XGBoost for regression tasks.

4. **Testing**: The best-performing classifiers/regressors determined in Step 3 are tested using the testing data. We use various evaluation metrics for rigorous evaluations as reported earlier.

The fourth step is repeated five times for all datasets. Then the average score is calculated for each method and evaluation metric.

To assess diversity, we employ coverage as described in Naeem et al. (2020). Coverage is defined as the proportion of real records that have at least one fake record among their neighboring instances, which are determined using the k-nearest neighbors (k-NN) algorithm. In Fig. 7, we present an illustrative example of coverage calculation with k = 1. Coverage of the real data is theoretically 1. We determine the hyperparameter k following the methodology outlined in the original paper, where the coverage of the real data exceeds 0.95. The reported coverage scores are the averaged scores after evaluate five times as well.

Detailed metrics for our experiments are as follows:

1. Binary F1 for binary classification datasets: `f1_score` from `sklearn.metrics` after setting the 'average' option to 'binary'.
2. Macro F1 for multi-class classification datasets: `f1_score` from `sklearn.metrics` after setting the 'average' option to 'macro'.
3. AUROC: `roc_auc_score` from `sklearn.metrics`.
4. R^2: `r2_score` from `sklearn.metrics`.
5. RMSE: `mean_squared_error` from `sklearn.metrics` after setting the 'squared' option to 'False'.
6. Coverage: `compute_prdc` from https://github.com/clovaai/generative-evaluation-prdc.

B.4 Hyperparameter settings for GADGET

Hyperparameter settings for the best models are summarized in Table 7. We have three SDE types, which are VE, VP, and sub-VP. We search for K in {2, 3, 5}, and use a learning rate in {1e-04, 5e-04, 1e-03, 5e-03}. We use γ_{min} in $\{0.01, 0.1, 1.0\}$ and γ_{max} in $\{5.0, 10.0\}$, which are parameters for $\gamma(t) = \gamma_{min} + t\,(\gamma_{max} - \gamma_{min})$ for $t \in [0, 1]$.

For the fine-tuning process, we consider a fine-tune learning rate in $\{i \times 10^{-j} | i = \{1, 2, 5\}, j = \{4, 5, 6, 7, 8, 9\}\}$.

Table 7: The best hyperparameters used for our experiments

Datasets	K	**Learning rate**	**SDE type**	γ_{min}	γ_{max}
Absenteeism	5	1E-03	sub-VP	0.01	5.0
Bank	3	5E-04	sub-VP	0.01	5.0
Clave	3	1E-03	VP	0.1	5.0
Concrete	2	5E-03	sub-VP	0.1	5.0
Contraceptive	2	1E-03	VP	0.01	5.0
Customer	3	1E-04	VP	0.1	10.0
Fish	2	5E-03	sub-VP	0.1	10.0
Heart Disease	3	5E-03	sub-VP	0.01	5.0
Nursery	5	5E-04	VP	1.0	5.0
Obesity	2	1E-03	VP	0.01	5.0
Spambase	2	1E-03	sub-VP	0.1	5.0

C Additional experimental results

We provide full experimental results in terms of the generative learning trilemma. We measure the metrics five times with different fake data by each method and report their mean and standard deviation.

C.1 Sampling quality

Table 8: Full experimental results in terms of the sampling quality for binary classification datasets. The best result is highlighted in bold face.

Methods	Bank		Heart Disease		Spambase	
	Binary F1	AUROC	Binary F1	AUROC	Binary F1	AUROC
Identity	0.520±0.038	0.895±0.080	0.972±0.057	0.990±0.025	0.935±0.026	0.983±0.017
MedGAN	0.000±0.000	0.500±0.000	0.594±0.067	0.659±0.061	0.000±0.000	0.500±0.000
VEEGAN	0.206±0.015	0.521±0.061	0.526±0.126	0.453±0.114	0.464±0.174	0.421±0.130
CTGAN	0.515±0.029	0.883±0.022	0.611±0.015	0.543±0.040	0.778±0.027	0.898±0.039
CTGAN-GADGET	0.533±0.012	0.874±0.033	0.797±0.029	0.884±0.052	0.793±0.061	0.920±0.054
TVAE	0.524±0.019	0.870±0.026	0.745±0.056	0.843±0.069	0.743±0.026	0.887±0.060
TableGAN	0.444±0.060	0.812±0.074	0.767±0.023	0.868±0.039	0.758±0.076	0.902±0.082
OCT-GAN	0.539±0.019	0.887±0.020	0.649±0.033	0.733±0.056	0.850±0.030	0.943±0.044
RNODE	0.263±0.043	0.739±0.083	0.815±0.036	0.894±0.055	0.833±0.049	0.935±0.030
RNODE-GADGET	0.193±0.128	0.780±0.099	0.823±0.029	0.901±0.030	0.839±0.067	0.941±0.051
SGM	0.562±0.041	0.907±0.030	0.838±0.033	0.926±0.025	0.890±0.031	0.964±0.026
STaSy	0.562±0.022	0.905±0.017	0.835±0.031	0.918±0.025	0.901±0.028	0.969±0.024
SGM-GADGET	0.567±0.031	0.915±0.019	0.875±0.036	0.936±0.047	0.901±0.024	0.969±0.018

Table 9: Full experimental results in terms of the sampling quality for multiclass classification datasets. The best result is highlighted in bold face.

Methods	Clave		Contraceptive		Customer		Nursery		Obesity	
	Macro F1	AUROC	Macro F1	AUROC	Macro F1	AUROC	Macro F1	AUROC	Macro F1	AUROC
Identity	0.703±0.099	0.916±0.059	0.466±0.016	0.679±0.020	0.370±0.022	0.669±0.021	0.795±0.005	0.999±0.002	0.966±0.013	0.997±0.004
MedGAN	0.245±0.031	0.647±0.022	0.382±0.002	0.590±0.011	0.210±0.043	0.493±0.030	0.097±0.000	0.500±0.000	0.040±0.000	0.500±0.000
VEEGAN	0.144±0.000	0.500±0.000	0.175±0.000	0.500±0.000	0.105±0.000	0.500±0.000	0.097±0.000	0.500±0.000	0.040±0.000	0.500±0.000
CTGAN	0.483±0.040	0.766±0.044	0.304±0.013	0.497±0.007	0.247±0.013	0.521±0.011	0.526±0.828	0.828±0.022	0.137±0.004	0.508±0.007
CTGAN-GADGET	0.489±0.066	0.790±0.077	0.413±0.018	0.611±0.025	0.289±0.023	0.573±0.032	0.541±0.026	0.877±0.030	0.506±0.037	0.872±0.017
TVAE	0.524±0.050	0.812±0.055	0.383±0.006	0.593±0.009	0.140±0.006	0.518±0.005	0.097±0.000	0.500±0.000	0.454±0.042	0.840±0.017
TableGAN	0.331±0.008	0.590±0.007	0.343±0.011	0.577±0.004	0.241±0.008	0.512±0.005	0.294±0.010	0.697±0.029	0.316±0.028	0.782±0.007
OCT-GAN	0.498±0.047	0.791±0.057	0.390±0.011	0.594±0.012	0.227±0.015	0.493±0.003	0.587±0.009	0.866±0.021	0.445±0.024	0.797±0.018
RNODE	0.449±0.029	0.764±0.039	0.419±0.026	0.625±0.029	0.314±0.018	0.610±0.038	0.293±0.024	0.599±0.030	0.461±0.024	0.846±0.022
RNODE-GADGET	0.481±0.033	0.776±0.040	0.410±0.037	0.641±0.026	0.319±0.027	0.604±0.048	0.350±0.029	0.670±0.047	0.485±0.056	0.862±0.024
SGM	0.595±0.072	0.879±0.066	0.427±0.012	0.639±0.028	0.310±0.015	0.616±0.026	0.482±0.024	0.827±0.010	0.878±0.028	0.983±0.015
STaSy	0.587±0.069	0.871±0.072	0.447±0.023	0.655±0.028	0.338±0.037	0.635±0.044	0.566±0.035	0.862±0.011	0.905±0.025	0.984±0.019
SGM-GADGET	0.607±0.080	0.863±0.061	0.447±0.009	0.645±0.034	0.362±0.020	0.637±0.042	0.719±0.016	0.851±0.040	0.917±0.032	0.982±0.026

Table 10: Full experimental results in terms of the sampling quality for regression datasets. The best result is highlighted in bold face.

Methods	Absenteeism		Concrete		Fish	
	R^2	RMSE	R^2	RMSE	R^2	RMSE
Identity	0.385±0.028	0.775±0.018	0.900±0.048	0.181±0.044	0.610±0.019	0.291±0.007
MedGAN	-4.289±3.011	2.215±0.618	-0.027±0.075	0.589±0.022	-0.151±0.126	0.497±0.027
VEEGAN	-5.413±5.694	2.327±1.128	-2.710±2.218	1.065±0.335	-0.228±0.163	0.508±0.028
CTGAN	-0.063±0.047	1.018±0.022	0.063±0.052	0.562±0.014	0.011±0.128	0.462±0.030
CTGAN-GADGET	-0.068±0.081	1.020±0.038	0.318±0.046	0.480±0.016	0.397±0.066	0.361±0.020
TVAE	-0.960±0.131	1.383±0.045	0.207±0.132	0.516±0.041	0.214±0.086	0.411±0.021
TableGAN	-0.292±0.245	1.119±0.109	0.388±0.073	0.454±0.026	0.527±0.020	0.320±0.007
OCT-GAN	-0.042±0.041	1.009±0.020	0.017±0.025	0.576±0.008	0.129±0.125	0.433±0.031
RNODE	0.016±0.029	0.980±0.015	0.783±0.009	0.271±0.005	0.532±0.010	0.318±0.004
RNODE-GADGET	0.783±0.016	0.775±0.003	0.389±0.002	0.656±0.015	0.548±0.011	0.313±0.004
SGM	0.036±0.060	0.969±0.030	-0.205±0.641	0.612±0.186	0.396±0.023	0.361±0.007
STaSy	0.016±0.085	0.980±0.042	0.833±0.011	0.238±0.008	0.574±0.017	0.304±0.006
SGM-GADGET	0.167±0.027	0.902±0.015	0.835±0.022	0.236±0.016	0.548±0.005	0.313±0.002

C.2 SAMPLING DIVERSITY

Table 11: Full experimental results in terms of the sampling diversity for binary classification dataset. We use coverage to evaluate the diversity of the generated datasets. The best result is highlighted in bold face.

Methods	Bank	Heart Disease	Spambase
MedGAN	0.000±0.000	0.090±0.011	0.001±0.000
VEEGAN	0.001±0.000	0.004±0.000	0.002±0.000
CTGAN	0.774±0.001	0.098±0.010	0.455±0.011
CTGAN-GADGET	0.605±0.000	0.192±0.014	0.265±0.004
TVAE	0.783±0.003	0.287±0.010	0.695±0.007
TableGAN	0.623±0.004	0.816±0.031	0.598±0.009
OCT-GAN	0.623±0.004	0.004±0.000	0.552±0.010
RNODE	0.403±0.003	0.780±0.017	0.281±0.006
RNODE-GADGET	0.566±0.002	0.675±0.027	0.296±0.006
SGM	0.702±0.003	0.772±0.023	0.347±0.003
STaSy	0.854±0.012	0.839±0.009	0.764±0.004
SGM-GADGET	0.865±0.003	0.895±0.014	0.727±0.008

Table 12: Full experimental results in terms of the sampling diversity for multiclass classification datasets. We use coverage to evaluate the diversity of the generated datasets. The best result is highlighted in bold face.

Methods	Clave	Contraceptive	Customer	Nursery	Obesity
MedGAN	0.094±0.003	0.563±0.015	0.015±0.001	0.079±0.001	0.000±0.000
VEEGAN	0.208±0.003	0.005±0.001	0.003±0.000	0.002±0.000	0.000±0.000
CTGAN	0.482±0.002	0.721±0.010	0.428±0.013	0.185±0.004	0.276±0.005
CTGAN-GADGET	0.709±0.003	0.587±0.020	0.398±0.010	0.468±0.003	0.248±0.007
TVAE	0.709±0.003	0.469±0.011	0.398±0.010	0.468±0.003	0.359±0.009
TableGAN	0.239±0.003	0.758±0.009	0.896±0.013	0.262±0.003	0.353±0.004
OCT-GAN	0.499±0.004	0.531±0.005	0.073±0.009	0.241±0.005	0.296±0.011
RNODE	0.656±0.002	0.678±0.014	0.721±0.020	0.244±0.002	0.303±0.008
RNODE-GADGET	0.641±0.005	0.755±0.014	0.621±0.018	0.267±0.004	0.329±0.012
SGM	0.768±0.005	0.768±0.005	0.443±0.017	0.255±0.004	0.591±0.020
STaSy	0.397±0.003	0.875±0.013	0.713±0.025	0.411±0.003	0.633±0.007
SGM-GADGET	0.715±0.003	0.897±0.009	0.745±0.016	0.523±0.003	0.765±0.004

Table 13: Full experimental results in terms of the sampling diversity for regression datasets. We use coverage to evaluate the diversity of the generated datasets. The best result is highlighted in bold face.

Methods	Absenteeism	Concrete	Fish
MedGAN	0.015±0.001	0.001±0.000	0.024±0.002
VEEGAN	0.000±0.000	0.000±0.000	0.000±0.000
CTGAN	0.266±0.016	0.214±0.025	0.449±0.016
CTGAN-GADGET	0.210±0.024	0.148±0.008	0.513±0.023
TVAE	0.119±0.017	0.122±0.012	0.343±0.014
TableGAN	0.262±0.022	0.318±0.015	0.654±0.026
OCT-GAN	0.026±0.006	0.274±0.020	0.396±0.017
RNODE	0.308±0.013	0.426±0.020	0.762±0.018
RNODE-GADGET	0.320±0.016	0.415±0.017	0.771±0.014
SGM	0.114±0.011	0.640±0.021	0.758±0.027
STaSy	0.308±0.010	0.758±0.015	0.926±0.010
SGM-GADGET	0.222±0.012	0.552±0.014	0.945±0.013

C.3 SAMPLING TIME

Table 14: Full experimental results in terms of the sampling time for binary classification datasets. The best result is highlighted in bold face.

Methods	Bank	Heart Disease	Spambase
MedGAN	0.0246±0.0001	0.0249±0.0002	0.0398±0.0017
VEEGAN	0.0331±0.0051	0.0271±0.0016	0.0449±0.0039
CTGAN	0.1900±0.0028	0.2479±0.0020	0.3071±0.0303
CTGAN-GADGET	0.1485±0.0005	0.1267±0.0021	0.0472±0.0007
TVAE	0.0261±0.0040	0.0208±0.0001	0.0259±0.0011
TableGAN	0.0253±0.0058	0.0110±0.0002	0.0252±0.0035
OCT-GAN	0.8224±0.0097	0.7671±0.0026	1.0986±0.0270
RNODE	79.1960±3.3833	14.4123±0.2339	18.5746±0.6254
RNODE-GADGET	19.4909±0.2691	30.2750±0.6096	14.1030±0.0744
SGM	4.7591±0.2777	107.6362±2.3226	5.0813±0.2084
STaSy	13.1132±0.1283	51.6194±0.4906	61.5235±0.7319
SGM-GADGET	10.3983±0.4351	5.2705±0.1995	2.8611±0.6832

Table 15: Full experimental results in terms of the sampling time for multiclass classification datasets. The best result is highlighted in bold face.

Methods	Clave	Contraceptive	Customer	Nursery	Obesity
MedGAN	0.0278±0.0046	0.0247±0.0000	0.0005±0.0005	0.0261±0.0040	0.0265±0.0046
VEEGAN	0.0285±0.0002	0.0218±0.0006	0.0002±0.0002	0.0251±0.0038	0.0261±0.0055
CTGAN	0.1793±0.0094	0.1692±0.0018	0.0104±0.0104	0.1761±0.0022	0.1767±0.0073
CTGAN-GADGET	0.1590±0.0005	0.0327±0.0003	0.0032±0.0032	0.0317±0.0041	0.1914±0.0028
TVAE	0.0212±0.0033	0.0188±0.0002	0.0032±0.0032	0.0317±0.0041	0.0210±0.0043
TableGAN	0.0170±0.0067	0.0086±0.0001	0.0001±0.0001	0.0153±0.0058	0.0145±0.0009
OCT-GAN	0.7642±0.0041	7.6209±0.0472	0.0135±0.0135	0.7931±0.0127	4.6496±0.0229
RNODE	11.4688±0.4156	10.8183±0.3609	7.4940±0.0327	10.5633±0.0193	7.0291±0.0544
RNODE-GADGET	11.7457±0.2876	34.1074±1.7643	6.3139±0.0293	14.9753±0.2831	15.8296±0.3052
SGM	16.3418±0.2134	5.1752±0.2526	7.7790±0.2061	2.3430±0.1065	5.2210±0.1218
STaSy	3.5149±0.2292	3.2419±0.2194	4.9578±0.2875	3.7901±0.1784	3.8308±0.0605
SGM-GADGET	2.2008±0.0319	3.5827±0.1936	1.6791±0.1684	3.4335±0.0824	2.6244±0.0568

Table 16: Full experimental results in terms of the sampling time for regression datasets. The best result is highlighted in bold face.

Methods	Absenteeism	Concrete	Fish
MedGAN	0.0004±0.0384	0.0261±0.0051	0.0236±0.0002
VEEGAN	0.0022±0.0316	0.0161±0.0001	0.0155±0.0003
CTGAN	0.0017±0.1986	0.0569±0.0002	0.0433±0.0002
CTGAN-GADGET	0.1223±0.0003	0.0119±0.0001	0.0184±0.0001
TVAE	0.0013±0.0224	0.0218±0.0041	0.0295±0.0001
TableGAN	0.0002±0.0145	0.0128±0.0060	0.0111±0.0002
OCT-GAN	0.0209±1.1692	0.6801±0.0053	0.4809±0.0039
RNODE	18.3608±0.1162	1.4792±0.0238	0.9925±0.0246
RNODE-GADGET	13.5653±0.2859	2.8808±0.0506	1.6716±0.0194
SGM	5.8162±0.4003	3.8166±0.1268	3.6888±0.1072
STaSy	21.4828±0.1382	1.3293±0.0496	1.2404±0.0055
SGM-GADGET	4.7351±0.1272	1.2531±0.0973	0.8905±0.0502

D SAMPLING COMPLEXITY OF SGM FOR TABULAR DATA VS. SGM-GADGET

Algorithm 2 *Probability flow* for SGM

Input Trained score model $F_{\boldsymbol{\theta}}$, final time $t_0 = 0$, start time $t_1 = 1$, and a Gaussian prior $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]$

Output Generated fake data $\hat{\mathcal{T}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_N]$

function `ode_func` $([\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N], t)$ **:**
  for $i \in \{1, 2, \ldots, N\}$ **do**
    $s_i \leftarrow F_{\boldsymbol{\theta}}(\mathbf{z}_i, t)$
    $\mathbf{r} \leftarrow \mathbf{r} \cup {}^{\mathrm{d}\mathbf{z}_i}/_{\mathrm{d}t}$, where ${}^{\mathrm{d}\mathbf{z}_i}/_{\mathrm{d}t} = \mathbf{f}(\mathbf{z}_i, t) - \frac{1}{2} g^2(t) s_i$
  return r
$[\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_N] \leftarrow$ odeint(`ode_func`, $[\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N], t_0, t_1$) **return** $\hat{\mathcal{T}}$

Algorithm 3 *Probability flow* for SGM-GADGET

Input Trained score models $F_{\boldsymbol{\theta}_k}$ for all k, final time $t_0 = 0$, start time $t_1 = 1$, and a Gaussian prior $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]$

Output Generated fake data $\hat{\mathcal{T}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_N]$

function `ode_func` $([\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N], t)$ **:**
  $i = 1$ **for** $k \in \{1, 2, \ldots, K\}$ **do**
    for $j \in \{1, 2, \ldots, |\mathcal{B}_k|\}$ **do**
      $s_i \leftarrow F_{\boldsymbol{\theta}_k}(\mathbf{z}_i, t)$
      $\mathbf{r} \leftarrow \mathbf{r} \cup {}^{\mathrm{d}\mathbf{z}_i}/_{\mathrm{d}t}$, where ${}^{\mathrm{d}\mathbf{z}_i}/_{\mathrm{d}t} = \mathbf{f}(\mathbf{z}_i, t) - \frac{1}{2} g^2(t) s_i$ // $\sum_{k=1}^{K} |\mathcal{B}_k| = N$
      i += 1
  return r
$[\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_N] \leftarrow$ odeint(`ode_func`, $[\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N], t_0, t_1$) **return** $\hat{\mathcal{T}}$

The sampling process for SGM and SGM-GADGET is illustrated in Algorithms 2 and 3, respectively. As noted earlier, both SGM and SGM-GADGET generate samples through the *probability flow*. The *probability flow* is as follows:

$$\mathrm{d}\mathbf{x} = \big(\mathbf{f}(\mathbf{x}, t) - \frac{1}{2} g^2(t) \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\big) \mathrm{d}t, \tag{8}$$

where $\mathbf{f}$ is the draft and g is the diffusion coefficient of $\mathbf{x}$, and $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is approximated by a score network $F_{\boldsymbol{\theta}}$ for the *probability flow*.

SGM-GADGET consists of K SGMs, and one might assume that the sampling complexity of SGM-GADGET is of the order K. However, it is important to note that i) SGM-GADGET is trained using the *probability flow*, where we generate fake records using the neural ordinary differential equation (NODE) based on $F_{\boldsymbol{\theta}_k}$, and ii) each SGM's size can be smaller since it learns a Gaussian-like distribution. By employing an augmented ODE, we are able to generate samples from K SGMs simultaneously. Therefore, the total complexity can be smaller than existing single SGM-based approaches, e.g., STaSy.

We augment the ODE function with many noisy vectors sampled from a Gaussian prior for both SGM and GADGET. The difference in the sampling process lies within the ODE function itself. In the case of SGM, we iterate over N inputs using a single SGM $F_{\boldsymbol{\theta}}$. On the other hand, for GADGET, we iterate over N inputs using K SGMs $F_{\boldsymbol{\theta}_k}$, where each SGM $F_{\boldsymbol{\theta}_k}$ evaluates the dynamics on $|\mathcal{B}_k|$ inputs. This process gives us theoretically the same sampling complexity as that of SGM.

As a result, we have faster sampling processes compared to SGM in many cases (cf. Tables 2 and 14 to 16), and the improvement is apparently from the reduction in the number of learnable parameters in score networks (cf. Appendix E).

E THE NUMBER OF LEARNABLE PARAMETERS

In Table 17, we present the total number of learnable parameters in neural networks for each method. For methods involving multiple neural networks, such as GAN-based methods, we report the sum of the learnable parameters across all networks. Tables 18, 19, 21 to 23 and 26 provide the number of learnable parameters for each network within each baseline method. RNODE and STaSy consist of only one neural network. For SGM-GADGET, the same neural network is used for each of the K neural networks so they have an equal number of learnable parameters. To have the number of learnable parameters in a single neural network for SGM-GADGET, simply divide the value presented in Table 17 by the corresponding value of K from Table 7.

As shown in Table 17, the total number of parameters of baselines tends to be proportional to the generation performance, i.e., the sampling quality and diversity — however, our method enhances the performance without increasing the total sum of model parameters. Specifically, STaSy contains the biggest number of parameters and shows the best performance in terms of the generation quality among baselines. Moreover, RNODE, which requires the biggest parameter numbers among the baselines except for the SGM-based methods, performs the best in terms of the sampling quality and the second best in terms of the sampling diversity, excluding the SGM-based methods. Other GAN-based methods are more overfitted to training samples with more parameters in many cases.

Table 17: The total number of learnable parameters in neural networks for each method. For SGM-GADGET, it is the sum of all those K SGMs' parameter numbers.

Methods	Absent.	Bank	Clave	Conc.	Contra.	Customer	Fish	Heart.	Nursery	Obesity	Spam.	**Average**
MedGAN	276,716	77,493	127,396	106,633	149,697	173,343	105,095	103,239	124,320	69,798	259,387	143,011
VEEGAN	171,405	143,190	76,229	120,618	149,346	74,304	173,992	152,424	190,017	76,999	146,268	134,072
CTGAN	885,285	700,642	306,229	153,269	287,919	203,012	232,367	705,393	400,673	355,295	1,154,885	489,543
CTGAN-GADGET	153,082	47,628	83,775	38,036	63,838	38,750	37,136	93,656	60,674	42,022	60,536	65,376
TVAE	100,296	238,430	67,272	208,104	219,412	72,690	254,300	228,664	247,104	231,748	368,472	203,317
TableGAN	464,067	1,714,563	431,299	36,739	69,507	40,707	139,011	139,011	40,707	1,649,027	1,845,635	597,298
OCT-GAN	1,571,847	1,448,402	4,709,195	4,742,583	4,688,299	4,679,376	4,676,593	4,798,996	4,768,835	4,827,135	1,341,977	3,841,203
RNODE	4,219,485	100,648	54,865	38,589	30,553	351,852	23,721	2,285,097	125,569	263,715	144,197	737,458
RNODE-GADGET	800,714	58,726	27,650	13,716	349,444	61,755	6,358	830,420	65,538	122,212	144,434	223,257
SGM	5,945,404	5,542,254	5,418,868	5,224,090	5,629,698	5,382,688	5,209,720	5,673,528	5,389,920	5,433,354	5,585,940	5,494,133 STaSy 5,941,820
5,542,254	5,418,868	5,220,506	5,629,698	5,382,688	5,206,136	5,673,528	5,389,920	5,433,354	5,585,940	5,493,156		
SGM-GADGET	2,823,660	11,245,962	123,804	219,316	938,372	115,104	50,288	11,584,488	381,600	1,952,660	328,616	2,705,806

Table 18: The number of learnable parameters in MedGAN for each network

Networks	Absenteeism	Bank	Clave	Concrete	Contraceptive	Customer	Fish	Heart Disease	Nursery	Obesity	Spambase
Encoder	27,904	6,912	4,736	1,280	8,448	8,192	1,024	9,216	4,224	4,992	15,360
Decoder	27,756	6,837	4,644	1,161	8,385	7,967	903	9,159	4,128	4,902	15,163
Discriminator	88,448	30,208	84,480	70,656	99,328	24,576	69,632	51,328	82,432	26,368	96,256
Generator	132,608	33,536	33,536	33,536	33,536	132,608	33,536	33,536	33,536	33,536	132,608
Total	276,716	77,493	127,396	106,633	149,697	173,343	105,095	103,239	124,320	69,798	259,387

Table 19: The number of learnable parameters in VEEGAN for each network

Networks	Absenteeism	Bank	Clave	Concrete	Contraceptive	Customer	Fish	Heart Disease	Nursery	Obesity	Spambase
Discriminator	102,145	88,065	25,473	76,801	91,137	24,833	76,289	92,673	82,689	25,729	89,601
Generator	34,668	27,573	25,380	21,897	29,121	24,735	76,039	29,895	82,464	25,638	28,347
Reconstructor	34,592	27,552	25,376	21,920	29,088	24,736	21,664	29,856	24,864	25,632	28,320
Total	171,405	143,190	76,229	120,618	149,346	74,304	173,992	152,424	190,017	76,999	146,268

Table 20: The number of learnable parameters in CTGAN for each network

Networks	Absenteeism	Bank	Clave	Concrete	Contraceptive	Customer	Fish	Heart Disease	Nursery	Obesity	Spambase
Discriminator	731,905	468,225	250,625	83,329	206,209	123,009	184,065	473,345	230,145	180,609	1,003,265
Generator	153,380	232,417	55,604	69,940	81,710	80,003	48,302	232,048	170,528	174,686	151,620
Total	885,285	700,642	306,229	153,269	287,919	203,012	232,367	705,393	400,673	355,295	1,154,885

Table 21: The number of learnable parameters in CTGAN-GADGET for each network

Networks	Absenteeism	Bank	Clave	Concrete	Contraceptive	Customer	Fish	Heart Disease	Nursery	Obesity	Spambase
Discriminator	21,313	4,417	13,121	9,089	5,729	3,745	8,961	16,897	12,609	4,001	12,289
Generator	55,228	19,397	14,804	9,929	26,190	15,630	9,607	29,931	17,728	17,010	17,979
K	2	2	3	2	2	2	2	2	2	2	2
Total	153,082	47,628	83,775	38,036	63,838	38,750	37,136	93,656	60,674	42,022	60,536

Table 22: The number of learnable parameters in TVAE for each network

Networks	Absenteeism	Bank	Clave	Concrete	Contraceptive	Customer	Fish	Heart Disease	Nursery	Obesity	Spambase
Encoder	54,144	127,360	37,760	112,256	117,888	40,448	143,616	122,496	140,032	124,032	192,128
Decoder	46,152	111,070	29,512	95,848	101,524	32,242	110,684	106,168	107,072	107,716	176,344
Total	100,296	238,430	67,272	208,104	219,412	72,690	254,300	228,664	247,104	231,748	368,472

Table 23: The number of learnable parameters in TableGAN for each network

Networks	Absenteeism	Bank	Clave	Concrete	Contraceptive	Customer	Fish	Heart Disease	Nursery	Obesity	Spambase
Classifier	132,993	528,129	132,993	1,409	1,409	2,817	2,817	2,817	2,817	528,129	528,129
Discriminator	132,993	528,129	132,993	1,409	1,409	2,817	2,817	2,817	2,817	528,129	528,129
Generator	198,081	658,305	165,313	33,921	66,689	35,073	133,377	133,377	35,073	592,769	789,377
Total	464,067	1,714,563	431,299	36,739	69,507	40,707	139,011	139,011	40,707	1,649,027	1,845,635

Table 24: The number of learnable parameters in OCT-GAN for each network

Networks	Absenteeism	Bank	Clave	Concrete	Contraceptive	Customer	Fish	Heart Disease	Nursery	Obesity	Spambase
Discriminator	1,167,745	1,154,561	4,548,353	4,543,233	4,567,809	4,551,169	4,541,697	4,570,625	4,546,305	4,562,689	1,181,313
Generator	404,102	293,841	160,842	199,350	120,490	128,207	134,896	228,371	222,530	264,446	160,664
Total	1,571,847	1,448,402	4,709,195	4,742,583	4,688,299	4,679,376	4,676,593	4,798,996	4,768,835	4,827,135	1,341,977

Table 25: The number of learnable parameters in RNODE for each network

Networks	Absenteeism	Bank	Clave	Concrete	Contraceptive	Customer	Fish	Heart Disease	Nursery	Obesity	Spambase
RNODE	4,219,485	100,648	54,865	38,589	30,553	351,852	23,721	2,285,097	125,569	263,715	144,197

Table 26: The number of learnable parameters in RNODE-GADGET for each network

Networks	Absenteeism	Bank	Clave	Concrete	Contraceptive	Customer	Fish	Heart Disease	Nursery	Obesity	Spambase
RNODE	400,357	29,363	13,825	6,858	174,722	20,585	3,179	415,210	32,769	61,106	72,217
K	3	2	2	2	2	2	2	5	2	2	5
Total	1,201,071	58,726	27,650	13,716	349,444	41,170	6,358	2,076,050	65,538	122,212	361,085