SOS: Score-based Oversampling for Tabular Data

Jayyoung Kim
Chaejeong Lee
tyehjin.shin@gmail.com
Yonsei University
South Korea
seo0413.park@samsung.com
Samsung SDS
South Korea

Yehjin Shin
mij001.kim@samsung.com
Samsung SDS
South Korea

Sewon Park
noseong@yonsei.ac.kr
Yonsei University
South Korea

Noseong Park
noseong@yonsei.ac.kr
Yonsei University
South Korea

Jihoon Cho
sw0413.park@samsung.com
Samsung SDS
South Korea

ABSTRACT

Score-based generative models (SGMs) are a recent breakthrough in generating fake images. SGMs are known to surpass other generative models, e.g., generative adversarial networks (GANs) and variational autoencoders (VAEs). Being inspired by their big success, in this work, we fully customize them for generating fake tabular data. In particular, we are interested in oversampling minor classes since imbalanced classes frequently lead to sub-optimal training outcomes. To our knowledge, we are the first presenting a score-based tabular data oversampling method. Firstly, we re-design our own score network since we have to process tabular data. Secondly, we propose two options for our generation method: the former is equivalent to a style transfer for tabular data and the latter uses the standard generative policy of SGMs. Lastly, we define a fine-tuning method, which further enhances the oversampling quality. In our experiments with 6 datasets and 10 baselines, our method outperforms other oversampling methods in all cases.

CCS CONCEPTS
• Computing methodologies → Artificial intelligence.

KEYWORDS
score-based generative model; tabular data synthesis; oversampling

1 INTRODUCTION

Tabular data is one of the most frequently occurring data types in the field of data mining and machine learning. However, imbalanced situations happen from time to time, e.g., one class significantly outweighs other classes. Oversampling minor classes (and as a result, making class sizes even) is a long-standing research topic. Many methods have been proposed, from classical statistical methods [7, 19, 20], to recent deep generative methods [27, 28].

Among many deep generative models, score-based generative models (SGMs [36]) have gained much attention recently. They adopt a stochastic differential equation (SDE) based framework to model i) the (forward) diffusion process and ii) the (reverse) denoising process (cf. Fig. 1 (a)). The diffusion process is to add increasing noises to the original data \( x_0 \), and the denoising process is the reverse of the diffusion process to remove noises from the noisy data \( x_T \). Since the space of \( x \) can be tremendously large, the score function, \( \nabla_x \log p_t(x) \) is approximated by a score neural network (or simply a score network), where \( t \in [0, T] \) and the step sizes of the diffusion and the denoising processes are typically 1. The ground-truth score values are collected during the forward SDE process. In comparison with generative adversarial networks (GANs), SGMs are known to be superior in terms of generation quality [12, 38, 43]. However, one drawback of SGMs is that they require large-scale computation, compared to other generative models. The main reason is that SGMs sometimes require long-step SDEs, e.g., \( T = 1,000 \) for images in [38]. It takes long time to conduct the denoising process with such a large number of steps. At the same time, however, it is also known that small \( T \) values work well for time series [34].

In this paper, we propose a novel Score-based Oversampling (SOS) method for tabular data and found that i) \( T \) can be small in our case because of our model design, ii) the score network architecture should be significantly redesigned, and iii) a fine-tuning method, inspired by one characteristic of imbalanced tabular data that we have both target and non-target classes, can further enhance the synthesis quality.

As shown in Fig. 1 (b), we separately train two SGMs, one for the non-target (or major) class and the other for the target (or minor) class to oversample — for simplicity but without loss of generality, we assume the binary class data, but our model is such flexible that multiple minority classes can be augmented. In other words, we separately collect the ground-truth score values of the target and non-target classes and train the score networks. Therefore, their
We note that it means the gradient of the log probability w.r.t. x at time t.

\[ \frac{dx}{dt} = f(x, t)dt + g(t)dw \]

(a) The overall workflow of score-based generative models, where the score function is approximated by a score network, i.e., \( S_0(x, t) = \nabla_x \log p_t(x) \).

We note that it means the gradient of the log probability w.r.t. x at time t.

\[ \frac{dx}{dt} = (f(x, t) - g^2(t)\nabla_x \log p_t(x))dt + g(t)dw \]

(b) The proposed concept where i) we transfer a non-target (or major) record \( x_0^f \) to a synthesized target (or minor) record \( x_0^g \) or ii) we sample a noisy vector \( z \) and generate \( x_0^g \).

(c) The proposed fine-tuning method where we i) compare the two gradients, and ii) fine-tune \( \theta \) to enhance the sampling quality.

Figure 1: In this figure, we assume one major (non-target) and one minor (target) classes for ease of discussion but our propose method can be applied to multiple minor classes. (a) The reverse SDE can be considered as a generator since it produces data from noise. (b) The combination of the major (non-target) class’s forward SDE and the minor (target) class’s reverse SDE can be considered as a style transfer-based oversampling process. (c) The fine-tuning process can further enhance the minor (target) class’s score network.

reverse SDEs will be very different. After that, we have two model designs denoted as Option 1 and 2 in Fig. 1 (b). By combining the non-target class’s forward SDE and the target class’s reverse SDE, firstly, we can create a style transfer. A non-target record \( x_0^f \) is converted to a noisy record \( x_0^f \), from which a fake target record \( x_0^g \) is created. In other words, the style (or class label) of \( x_0^f \) is transferred to that of the target class. Secondly, we sample a noisy vector \( z \) and generate a fake target record.

Since the forward SDE is a pure Brownian motion process shared by the non-target and the target classes, it constitutes a single noisy space. Since we maintain a single noisy space for both of them, \( x_0^f \) is a correct noisy representation of the non-target record \( x_0^f \) even when it comes to the target class’s reverse SDE. Therefore, the first option in Fig. 1 (b) can be considered as a style transfer-based oversampling method. Its main motivation is to oversample around class boundaries since the non-target record is (little) perturbed to generate the fake target record (cf. Fig. 6).

We further enhance our method by adopting the fine-tuning method in Fig. 1 (c), a post-processing procedure after training the two score networks. Our fine-tuning concept is to compare the two gradients (approximated) by \( S_0^f \) and \( S_0^g \) at a certain \((x_t, t = \epsilon_t)\) pair, where \( x \in \mathcal{T} \) is a real record, \( \mathcal{T} \) is tabular data, and \( \epsilon_t \) is a time point close to 0 — note that at this step, we do not care about the class label of x but use all records in the tabular data. If the angle between the two gradients is small, we regulate \( S_0^f \) to discourage the sampling direction toward the measured gradient (since the direction will lead us to a region where the non-target and target records co-exist).

We conduct experiments with 6 benchmark tabular datasets and 10 baselines: 4 out of 6 datasets are for binary classification and 2 are for multi-class classification. Our careful choices of the baselines include various types of oversampling algorithms, ranging from classical statistical methods, generative adversarial networks (GANs), variational autoencoders (VAEs) to neural ordinary differential equation-based GANs. In those experiments, our proposed method significantly outperforms existing methods by large margins — in particular, our proposed method is the first one increasing the test score commonly for all those benchmark datasets. Our contributions can be summarized as follows:

1. We, for the first time, design a score-based generative model (SGM) for oversampling tabular data. Existing SGMs mainly study image synthesis.
2. We design a neural network to approximate the score function for tabular data and propose a style transfer-based oversampling technique.
3. Owing to one characteristic of tabular data that we always have class labels for classification, we design a fine-tuning method, which further enhances the sampling quality.
4. In our oversampling experiments with 6 benchmark datasets and 10 baselines, our method not only outperforms them but also increases the test score for all cases in comparison with the test score before oversampling (whereas existing methods fail in at least one case).
5. In Appendix E, our method overwhelms baselines when generating full fake tabular data (instead of oversampling minor classes) as well.

2 RELATED WORK AND PRELIMINARIES

In this section, we review papers related to our research. We first review the recent score-based generative models, followed by tabular data synthesis.

2.1 Score-based Generative Models

SGMs use the following Itô SDE to define diffusive processes:

\[ dx = f(x, t)dt + g(t)dw, \]  \hspace{1cm} (1)

where \( f(x, t) = f(t)x \) and its reverse SDE is defined as follows:

\[ dx = (f(x, t) - g^2(t)\nabla_x \log p_t(x))dt + g(t)dw \]  \hspace{1cm} (2)

where this reverse SDE process is a generative process (cf. Fig. 1 (a)). According to the types of \( f \) and \( g \), SGMs can be further classified as i) variance exploding (VE), ii) variance preserving (VP), and ii) sub-variance preserving (Sub-VP) models [38].

In general, \( x_0 \) means a real (image) sample and following the (3) Owing to one characteristic of tabular data that we always have class labels for classification, we design a fine-tuning method, which further enhances the sampling quality.

(4) In our oversampling experiments with 6 benchmark datasets and 10 baselines, our method not only outperforms them but also increases the test score for all cases in comparison with the test score before oversampling (whereas existing methods fail in at least one case).

(5) In Appendix E, our method overwhelms baselines when generating full fake tabular data (instead of oversampling minor classes) as well.
Multiple Generators

O (Score Networks)

GAN

Domain

Classical Method

N/A

Image

Tabular

Classical Method

N/A

Image

Table 1: The qualitative comparison among our method and other methods. There are no existing minor-class oversampling models that adopt the score-based generative model. Moreover, many of the existing minor-class oversampling methods are for images. In our experiments, we modify some of them to be able to process tabular data and compare with our method.

| Method     | Multiple Generators | Generative Model Type | Domain     |
|------------|---------------------|-----------------------|------------|
| SMOTE      | N/A                 | Classical Method      | Tabular    |
| B-SMOTE    | N/A                 | Classical Method      | Tabular    |
| ADASYN     | N/A                 | Classical Method      | Tabular    |
| RAN        | X                   | GAN                   | Image      |
| WGAN       | X                   | GAN                   | Tabular    |
| GL-GAN     | X                   | GAN                   | Tabular    |
| GAMO       | O                   | GAN                   | Image      |
| SOS (Ours) | O (Score Networks)  | Score-based Model     | Tabular    |

The transition probability (or the perturbation kernel) \( p(\mathbf{x}_t|\mathbf{x}_0) \) at time \( t \) with a Gaussian distribution. The reverse SDE in Eq. (2) maps a noisy sample at \( t = T \) to a data sample at \( t = 0 \).

We can collect the gradient of the log transition probability, \( \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0) \), during the forward SDE since it can be analytically calculated. Therefore, we can easily train a score network \( S_\theta : \mathbb{R}^{\text{dim}(\mathbf{x})} \times [0, T] \to \mathbb{R}^{\text{dim}(\mathbf{x})} \) as follows, where \( \text{dim}(\mathbf{x}) \) means the dimensionality of \( \mathbf{x} \), e.g., the number of pixels in the case of images or the number of columns in the case of tabular data:

\[
\arg \min_{\theta} \mathbb{E}_{\mathbf{x}_t, \mathbf{x}_0} \left[ \left( \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0) \right)^2 \right],
\]

where \( \lambda(t) \) is to control the trade-off between synthesis quality and likelihood. This training method is known as denoising score matching and it is known that \( \theta^* \) solving Eq. (3) can correctly solve the reverse SDE in Eq. (2). In other words, the optimal solution \( \theta^* \) of the denoising score matching training is equivalent to the optimal solution of the exact score matching training — its proof is in [41].

There exist several other improvements on SGMs. For instance, the adversarial score matching model [22] is to combine SGMs and GANs. Therefore, the reverse SDE is not solely determined as a reverse of the forward SDE but as a trade off between the denoising score matching and the adversarial training of GANs (cf. Fig. 2 (a)).

Once the training process is finished, one can synthesize fake samples by training a generator and a discriminator in an adversarial way [16]. Among many variants of GANs [3, 4, 8, 18, 30, 31], WGAN is widely used providing a more stable training of GANs [4]. Especially, WGAN-GP is one of the most sophisticated model and is defined as follows [18]:

\[
\min_G \max_D \mathbb{E}_{p_d}[D(\mathbf{x})] - \mathbb{E}_{p_z}[D(G(z))] - \lambda \mathbb{E}_{p_x}[\|\nabla \log D(\mathbf{x})\|_2^2],
\]

where \( G \) is a generator function, \( D \) is a Wasserstein critic function, \( p_\mathbf{z} \) is a prior distribution for a noisy vector \( z \), which is typically \( \mathcal{N}(0, 1) \), \( p_d \) is a data distribution, \( \hat{x} \) is a randomly weighted combination of \( G(z) \) and \( \mathbf{x} \), and \( \lambda \) is a regularization weight.

Tabular data synthesis generates realistic table records by modelling a joint probability distribution of features in a table. CLBN [10] is a Bayesian network built by the Chow-Liu algorithm representing a joint probability distribution. TVAE [44] is a variational autoencoder that effectively handles mixed types of features in tabular data. There are also many variants of GANs for tabular data synthesis. RGAN [15] generates continuous time-series records while MedGAN [9] generates discrete medical records using non-adversarial loss. TabLeGAN [32] generates tabular data using convolutional neural networks. PATE–GAN [23] was proposed to synthesize differentially private tabular records.

However, one notorious challenge in synthesizing tabular data with GANs is addressing the issue of mode collapse. In many cases, the fake data distribution synthesized by GANs is confined to only a few modes and fails to represent many others. Some variants of GANs tried to resolve this problem, proposing to force the generator to produce samples from the various modes as evenly as possible. VEEGAN [39] has a reconstructor network, which maps the output of the generator to the noisy vector space, and jointly trains the generator and the reconstructor network. CTGAN [44] has a conditional generator and use a mode separation process. OCT–GAN [24] exploits the homeomorphic characteristic of neural ordinary differential equations, when designing its generator, and now shows the state-of-the-art synthesis quality for many tabular datasets. However, one downside is that it requires a much longer training time than other models.

2.2 Tabular Data Synthesis

Generative adversarial networks (GANs) are the most popular deep learning-based model for tabular data synthesis, and they generate
We describe our proposed score-based oversampling method. We oversampling of classes \( \{C_m\}_{m=1}^M \), for which we aim at oversampling. (a) We randomly choose \( x_0^m \) from other non-target classes \( \{C_m\}_{m=2}^M \). (b) We randomly sample a noisy vector \( z \).

![Diagram](image)

(a) Option 1: Style transfer-based oversampling which corresponds to oversampling around class boundary

\[
N(\mu, \sigma^2) \sim z \xrightarrow{\text{Reverse SDE}} \hat{x}_0
\]

(b) Option 2: Plain score-based oversampling

Figure 3: Another diagram showing our model architectures. In this figure, we assume multiple minor classes. Given a set of classes \( \{C_m\}_{m=1}^M \), suppose that a target minor class is \( C_1 \), for which we aim at oversampling. We randomly choose \( x_0^m \) from other non-target classes \( \{C_m\}_{m=2}^M \). (b) We randomly sample a noisy vector \( z \).

2.3 Minor-class Oversampling

Multi-class data commonly suffers from imbalanced classes, and two different approaches have been proposed to address the issue; classical statistical methods and deep learning-based methods.

Classical statistical methods are mostly based on the synthetic minority oversampling technique (SMOTE [7]) method, which generates synthetic samples by interpolating two minor samples. Borderline SMOTE (B-SMOTE [19]) improves SMOTE to consider only the minor samples on the class boundary where the major samples occupy more than half of the neighbors. The adaptive synthetic sampling (ADASYN [20]) applies SMOTE on the minority samples close to the major samples.

Deep learning-based methods, based on GANs, mostly aim at oversampling minor class in images. In ACGAN [29], a generator generates all classes including the minor ones, and a discriminator returns not only whether the input is fake or real but also which class it belongs to. Inspired by ACGAN, BAGAN [27] constructs the discriminator which distinguishes the fake from the rest of the classes at once, showing good performance for minor-class image generation. cWGAN [14] is a WGAN-GP based tabular data oversampling model, which is equipped with an auxiliary classifier to consider the downstream classification task. GL-GAN [42] is a GAN that employs an autoencoder and SMOTE to learn and oversample the latent vectors and includes an auxiliary classifier to provide label information to the networks. GAMO [28] uses multiple generators, each of which is responsible for generating a specific class of data, and two separate critics, one for the real/fake classification and another for the entire class classification.

3 PROPOSED METHOD

We describe our proposed score-based oversampling method. We first sketch the overall workflow of our method, followed by its detailed designs.

Definition 3.1. Let \( T \) be a table (or tabular data) consisting of \( M \) classes, \( \{C_m\}_{m=1}^M \), where \( C_m \) is \( m \)-th class (subset) of \( T \), i.e., classes are disjoint subsets of \( T \). We define a minor class as \( C_j \) whose cardinality is smaller than the maximum cardinality, i.e., \( |C_j| < \max_{m} |C_m| \). In general, therefore, we have one major (with the largest cardinality) and multiple minor classes in \( T \). Our task is to oversample those minor classes, i.e., add artificial fake minor records, until \( |C_j| = \max_{m} |C_m| \) for all \( j \).

3.1 Overall Workflow

Our proposed score-based oversampling, shown in Fig. 3, is done by the following workflow:

1. We train a score-based generative model for each class \( C_m \). Let \( S_{\theta_m} \) be a trained score network for \( C_m \).
2. Let \( C_j \) be a minor class to oversample, i.e., target class. We fine-tune \( S_{\theta_j} \) using all records of \( T \) as shown in Fig. 1 (c).
3. In order to oversample each target (minor) class \( C_j \), we use one of the following two options as shown in Fig. 3. We repeat this until \( |C_j| = \max_{m} |C_m| \).

(a) After feeding a record randomly chosen from all other non-target classes except \( C_j \), we derive its noisy vector. By using \( S_{\theta_j} \), we synthesize a fake target (minor) record for \( C_j \). (cf. Fig. 3 (a)).

(b) After sampling a noisy vector \( z \sim N(\mu, \sigma^2) \), we generate a fake target (minor) record (cf. Fig. 3 (b)).

4. After generating a fake record for the target class \( C_j \), we can apply the Langevin corrector, as noted in Fig. 2 (b). However, this step is optional and for some types of SGMs, we also do not use this step sometimes — see Appendix A for its detailed settings in each dataset for our experiments. For instance, the type of Sub-VP does not use this step for reducing computation in [38] since it shows good enough quality even without the corrector.

Definition 3.2. Let \( C_j \) be a target minor class, for which we aim at oversampling. \( x_0^j \in C_j \) is a target (minor) record in the target class.

Definition 3.3. Let \( C_j \) be a target minor class, for which we aim at oversampling. \( x_0^i \in C_i \), where \( i \neq j \), is a record in other non-targeted classes. In the case of binary classification, \( C_j \) is a minor class and \( C_i \) is a major class. In general, however, \( x_0^i \) means a record from other non-target classes. For ease of our convenience for writing, we use the symbol ‘+’ to denote all other non-targeted classes. Fig. 3 follows this notation.

In this process, our main contributions lie in i) designing the score network \( S_{\theta_m} \) for each class \( C_m \), and ii) designing the fine-tuning method. Note that the fine-tuning happens after all score networks are trained.

Option 1 vs. Option 2. In Fig. 3, we show two options of how to sample a noisy vector. In Fig. 3 (a), we perturb \( x_0^m \) in a non-target class to generate \( \hat{x}_0^m \). This strategy can be considered as oversampling around class boundary. In Fig. 3 (b), we sample from random noisy vectors and therefore, there is no guarantee that \( \hat{x}_0^m \) is around class boundary. We refer to Fig. 6 for its intuitive visualization result.

One more advantage of the first option is that we do not require that \( x_0^+ \) follows a Gaussian distribution. Regardless of the distribution of \( x_0^+ \), we can derive \( \hat{x}_0^m \) using the reverse SDE with \( S_{\theta_m} \) —
recall that we do not sample the noisy vector from a Gaussian in the first option but from $x_t^j$ using the forward SDE. Therefore, we can freely choose a value for $T$, which is one big advantage in our design since a large $T$ increases computational overheads. For the second option, we also found that small $T$ settings work well as in [34].

The relationship between the first and the second option is analogous to that between Borderline1-line-SMOTE (or B-SMOTE) and SMOTE. In this work, we propose those two mechanisms for increasing the usability of our method.

3.2 Score-based Model for Tabular Data

We use the forward and reverse SDEs in Eqs. (1) and (2). As noted earlier, there exist three types of SGMs, depending on the types of $f$ and $g$ in Eqs. (1) and (2). We introduce the definitions of $f$ and $g$ in Appendix B and the proposed architecture for our score network in Appendix C.

Since the original score network design is for images, we redesign it with various techniques used by tabular data synthesis models [17, 25, 44]. Its details are in Appendix C. However, we train this score network using the denoising score matching loss as in the original design.

Our method provides two options for oversampling $C_j$. In the first option, we adopt the style transfer-based architecture, where i) we choose $x_t^j$ from $C_i$, $i \neq j$, ii) we derive a noisy vector from it using the forward SDE, and iii) we finally transfer the noisy vector to a fake target record. The rationale behind this design is that the forward SDE is a pure Brownian motion regardless of classes— in other words, all classes share the same forward SDE and thereby, share the same noisy space as well. Therefore, $x_t^j$ contains small information on its original record $x_t^j \in C_i$, which is still effective when it comes to generating a fake target record for $C_j$. Thus, this option can be considered as oversampling around class boundary.

In the second option, we follow the standard use of SGMs, where we sample $z \sim \mathcal{N}(\mu, \sigma^2)$ and generate a fake target record. We note that the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ is defined by the types of SDEs, i.e., VE, VP, or Sub-VP, as follows:

$$z \sim \begin{cases} 
\mathcal{N}(0, \sigma_{\text{max}}^2), & \text{if VE}, \\
\mathcal{N}(0, 1), & \text{if VP}, \\
\mathcal{N}(0, 1), & \text{if Sub-VP},
\end{cases}$$

(4) where $\sigma_{\text{max}}$ is a hyperparameter.

3.3 Fine-tuning Method

Once score networks $\{S_{\theta_m}\}_{m=1}^M$ are trained, we can sample fake records for all target classes as described earlier. In the case of tabular data, however, we can exploit class information to further enhance the sampling quality. Our proposed fine-tuning procedures are as follows— for simplicity, we describe how to fine-tune for $C_j$ only. However, we repeat the following method for each $j$:

(1) Firstly, given a record $x \in \mathcal{T}$ regardless of its class label and a chosen time $t = \epsilon_t^j$, we evaluate with each score network $S_{\theta_m}$ to know the gradient of the log probability w.r.t. $x_t$ at $x_t$.

3.4 Training Algorithm

Algorithm 1 shows the overall training process for our method. At the beginning, we initialize the score network parameters for all $m$. We then train each score network $S_{\theta_m}$ using each $C_m$ in Line 3. At this step, we use the standard denoising score matching loss in Eq. (3). After finishing this step, we can already generate fake target records. However, we further enhance each score network in Line 7. The complexity of the proposed fine-tuning is $O(|\mathcal{T}|m)$, which is affordable in practice since $m$, the number of classes, is not large in comparison with the table size $|\mathcal{T}|$. For oversampling $C_j$, we can use either the first or the second option of our model.
We describe the experimental environments and results. In Appendix E, we introduce our additional experiments that we generate entire fake tabular data (instead of making imbalanced tabular data balanced after oversampling minor classes). We also analyze the space and time overheads of our method in Appendix D.

4 EXPERIMENTS

4.1 Experimental Environments

4.1.1 Datasets. We describe the tabular datasets that are used for our experiments. Buddy and Satimage are for multi-class classification, and the others for binary classification.

Default [26] is a dataset describing the information on credit card clients in Taiwan regarding default payments, demographic factors, credit data, and others. The 'default payment' column includes 23,364 (77.9%) of major records and 6,636 (22.1%) minor records. Shoppers [35] is a binary classification dataset, where out of 12,330 sessions, 10,422 (84.5%) are negative records which did not end with purchase, and the rest 1,908 (15.5%) are positive records ending with purchase. Surgical [37] is a binary classification dataset, which contains general surgical patient information. There are 10,945 (74.8%) records for the major class, and 3,690 (25.2%) records for the minor class. WeatherAUS [13] contains 10 years of daily weather observations from several weather stations in Australia. 'RainTomorrow' is the target variable to classify, where 43,993 (78.0%) records belong to the major class, which means the rain for that day was over 1mm, and 12,427 (22.0%) records belong to the minor class.

Buddy [2] consists of parameters, including a unique ID, assigned to each animal that is up for adoption, and a variety of physical attributes. The target variable is 'breed', which includes 10,643 (56.5%) records for the major class, and 7,021 (37.3%) and 1,336 (6.2%) records for minor classes. Satimage [13] is to classify types of land usage, which is created by the Australian Centre for Remote Sensing. The dataset contains 2,282 (35.8%) records for the major class, and 1,336 (21.0%), 1,080 (16.9%), 562 (8.8%), 573 (9.0%), and 539 (8.5%) records for minor classes.

4.1.2 Baselines. We use a set of baselines as follows, which include statistical methods and generative models based on deep learning:

- Identity is the original tabular data without oversampling.
- SMOTE [7], B-SMOTE [19], and AdaSyn [20] are classical methods of data augmentation for the minority class; MedGAN [9], VEEGAN [39], TableGAN [32], and CTGAN [44] are GAN models for tabular data synthesis; TVAE [44] is proposed to learn latent embedding in VAE by incorporating deep learning metric. BAGAN [27] is a type of GAN model for oversampling images, but we replace its generator and discriminator with ours for tabular data synthesis. OCT-GAN [24] is one of the state of the art model for tabular data synthesis.

We exclude cWGAN [14] and GL-GAN [42], since they do not support multiple minority cases and synthesis quality is also relatively lower than others in our preliminary experiments.

4.1.3 Hyperparameters. We refer to Appendix A for detailed hyperparameter settings. We also provide all our codes and data for reproducibility and one can easily reproduce our results.

4.1.4 Evaluation Methods. Let $C_j$ be a minor class, where $|C_j| < \max_{m} |C_m|$. As mentioned earlier, our task is to oversample $C_j$ until $|C_j| = \max_{m} |C_m|$ for all $J$. However, some baselines are not for oversampling minor classes but generating one entire fake tabular data, in which case we train such a baseline for each $C_j$ — in other words, one dedicated baseline model for each minor class. For our method, we follow the described process.

For evaluation, we first train various classification algorithms with the augmented (or oversampled) training data — we note that the augmented data is fully balanced after oversampling — including Decision Tree [6], Logistic Regression [11], AdaBoost [36], and MLP [5]. We then conduct classification with the testing data. We take the average of the results — the same evaluation scenario where $\frac{\text{test score}}{\text{minor class}}$ is proposed to learn latent embedding in VAE using the denoising score matching loss in Eq. (3)

$$\mathcal{L} = \frac{1}{|C_j|} \sum_{j=1}^{M} \mathbb{E}_{x \sim C_j} \left[ \log \mathbb{P}(y | x) \right]$$

Fine-tune $\theta_j$ with $L(x, t, j)$ in Eq. (6)

$$L(x, t, j) = \|x \cdot t - y \cdot j\|_2^2$$

Oversample fake minor records with Option 1 or 2 for $C_j$ until $|C_j| = \max_{m} |C_m|$.

return oversampled minor classes
Table 2: Experimental results (Weighted F1). The best results are in boldface — if tie, the winner is determined by comparing in the 4th decimal place.

| Methods | Single Minority | Multiple Minority |
|---------|-----------------|-------------------|
|         | Default | Shoppers | Surgical | WeatherAUS | Buddy | Satimage |
| Identity | 0.515±0.035 | 0.601±0.039 | 0.687±0.004 | 0.657±0.016 | 0.603±0.010 | 0.817±0.004 |
| SMOTE   | 0.561±0.025 | 0.648±0.004 | 0.678±0.008 | 0.674±0.025 | 0.584±0.005 | 0.846±0.005 |
| B-SMOTE | 0.561±0.029 | 0.640±0.042 | 0.671±0.004 | 0.663±0.022 | 0.595±0.003 | 0.845±0.005 |
| Adasyn  | 0.558±0.023 | 0.630±0.045 | 0.662±0.007 | 0.658±0.022 | 0.608±0.002 | 0.841±0.008 |
| MedGAN  | 0.532±0.028 | 0.620±0.062 | 0.686±0.003 | 0.656±0.022 | 0.598±0.011 | 0.835±0.019 |
| VEEGAN  | 0.495±0.076 | 0.607±0.065 | 0.680±0.117 | 0.661±0.025 | 0.555±0.036 | 0.840±0.031 |
| TableGAN | 0.423±0.115 | 0.571±0.097 | 0.704±0.001 | 0.579±0.066 | 0.570±0.019 | 0.813±0.013 |
| TVAE    | 0.536±0.035 | 0.610±0.060 | 0.681±0.004 | 0.652±0.018 | 0.552±0.044 | 0.846±0.031 |
| CTGAN   | 0.545±0.022 | 0.605±0.059 | 0.701±0.004 | 0.659±0.020 | 0.593±0.009 | 0.833±0.015 |
| OCT–GAN | 0.531±0.018 | 0.639±0.029 | 0.692±0.082 | 0.656±0.018 | 0.551±0.015 | 0.837±0.011 |
| BANGLA  | 0.525±0.005 | 0.610±0.005 | 0.668±0.004 | 0.663±0.002 | 0.555±0.013 | 0.834±0.011 |

Table 3: Results by some selected key hyperparameters

| Hyper Param. | WeatherAUS | Satimage |
|--------------|------------|----------|
| Setting      | Weighted F1 | Setting | Weighted F1 |
| $\beta_{min}$ | 10 | 0.679±0.002 | 10 | 0.841±0.008 |
| $\beta_{max}$ | 50 | 0.680±0.003 | 50 | 0.857±0.006 |
| $\xi$ | 100 | 0.670±0.001 | 100 | 0.846±0.004 |
| $\omega$ | 150 | 0.660±0.002 | 150 | 0.847±0.005 |
| $\epsilon$ | 200 | 0.666±0.002 | 200 | 0.845±0.002 |
| $\epsilon_{t}$ | 300 | 0.669±0.002 | 300 | 0.849±0.004 |

4.3 Sensitivity on Hyperparameters

We summarize the sensitivity results w.r.t. some key hyperparameters in Table 3. We used $T = 50$ in Table 2. Others for $T$ produce sub-optimal results in Table 3. $\beta_{min}$ and $\beta_{max}$ means the lower and upper bounds of $\beta(t)$ of the drift $f$ and the diffusion $g$. In general, all settings produce reasonable outcomes, outperforming Identity. $\xi$ is the angle threshold for our fine-tuning method. As shown in Appendix A, the best setting for $\xi$ varies from one dataset to another but in general, $\xi \in [80, 100]$ produces the best fine-tuning results. For $\omega$, we recommend a value close to 0.99 as shown in Table 3. $\epsilon_{t} = 5e-04$ produces optimal results in almost all cases.

4.4 Sensitivity on Boundary vs. Regular Oversampling

The first option of our sampling strategy (cf. Fig. 3 (a)) corresponds to oversampling around class boundary, and the second option (cf. Fig. 3 (b)) corresponds to regular oversampling. We compare their key results in Table 4. It looks like that there does not exist a clear winner, but the boundary oversampling produces better results in more cases.

4.5 Ablation study on the reverse SDE solver (or predictor)

To solve $x_{t}^{r}$ from $x_{T}^{r}$ with Eq. (2), we can adopt various strategies. In general, the Euler-Maruyama (EM) method [33] is popular in solving SDEs. However, one can choose different SDE solvers, such as ancestral sampling [21], reverse diffusion [38], probability flow [38] and so on. Following the naming convention of [38], we call them as predictor (instead of solver) in this subsection. In the context of SGMs, therefore, the predictor means the SDE solver to solve Eq. (2).

In Table 5, the Euler-Maruyama method leads to the best outcomes for Surgical. For Shoppers, the ancestral sampling method produces the best outcomes. Likewise, predictors depend on datasets. The probability flow also shows reasonable results in our experiments. Especially, we observe that the probability flow always outperforms other predictors in Buddy.

4.6 Ablation study on Adversarial Score Matching vs. Fine-tuning

We also compare the two enhancement strategies, the adversarial score matching (cf. Fig. 2) and our fine-tuning, in Table 6. First of all,
we found that the adversarial score matching for tabular data is not as effective as that for image data. The original method of the adversarial score matching had been designed for images to integrate SGMs with GANs. We simply took the discriminator of CTGAN, one of the best performing GAN models for tabular data, and combine it with our model, when building the adversarial score matching for our tabular data. Although there can be a better design, the current implementation with the best existing discriminator model for tabular data does not show good performance in our experiments. In many cases, it is inferior to our method even without fine-tuning. However, our fine-tuning successfully improves in almost all cases.

4.7 Visualization

We introduce several key visualization results, ranging from histograms of columns to t-SNE plots.

4.7.1 Column-wise Histogram. Fig. 5 shows two histogram figures. The real histogram of values and the histogram by our method are similar to each other in both figures. However, CTGAN fails to capture the real distribution of the ‘bmi’ column of Surgical in Fig. 5 (a), which we consider the well-known mode-collapse problem of GANs, i.e., values in certain regions are more actively sampled than those in other regions. In Fig. 5 (b), CTGAN’s histogram shows three peaks whereas the real histogram has only one peak. Our method successfully captures it.

4.7.2 t-SNE plot of fake/real records. In Fig. 6, we visualize real and fake records — we use SOS (Sub-VP) with the boundary oversampling option in WeatherAUS. The translucent dots mean real records, and the solid blue dots are transferred to the solid red dots in the figure. As described earlier, those solid red dots are around the class boundary.

Table 5: Comparison between the boundary (Bnd.) vs. regular (Reg.) oversampling

| Predictor | WeatherAUS | Default | Satimage |
|-----------|------------|---------|----------|
|           | Pred. only | Pred. Corr. | Pred. only | Pred. Corr. | Pred. only | Pred. Corr. |
| EM        | 0.709±0.0002 | 0.709±0.003 | 0.710±0.004 | 0.712±0.002 |
| AS        | 0.706±0.002 | 0.707±0.005 | 0.706±0.004 | 0.711±0.002 |
| RD        | 0.701±0.003 | 0.708±0.003 | 0.707±0.002 | 0.711±0.001 |
| PF        | 0.704±0.004 | 0.712±0.002 | 0.705±0.005 | 0.712±0.002 |

Table 4: Comparison between the boundary (Bnd.) vs. regular (Reg.) oversampling

Table: Comparison between the boundary (Bnd.) vs. regular (Reg.) oversampling

| SDE Type | WeatherAUS | Default | Satimage |
|----------|------------|---------|----------|
| SOS (VE) | 0.670±0.002 | 0.569±0.001 | 0.571±0.002 | 0.854±0.004 | 0.854±0.002 |
| SOS (VP) | 0.673±0.002 | 0.559±0.005 | 0.557±0.004 | 0.855±0.005 | 0.857±0.006 |
| SOS (Sub-VP) | 0.680±0.003 | 0.574±0.003 | 0.574±0.002 | 0.854±0.005 | 0.855±0.004 |

5 CONCLUSIONS & LIMITATIONS

Oversampling minor classes is a long-standing research problem in data mining and machine learning. Many different methods have been proposed so far. In this paper, however, we presented a score-based oversampling method, called SOS. Since SGMs were originally invented for images, we carefully redesigned many parts of them. For instance, we designed a neural network to approximate the score function of the forward SDE and a fine-tuning method. In our experiments with 6 datasets and 10 baselines, our method overwhelms other oversampling methods in all cases. Moreover, our method does not decrease the F1 score after oversampling in all cases whereas existing methods fail to do so from time to time. From all these facts, we think that our method makes a great step forward in oversampling minor classes.
Even though SGMs perform well in general, they are still in an early phase when it comes to generating tabular data, and we hope that our work will bring much follow-up research. For instance, one can invent a drift and/or a diffusion function different from the three existing types and specialized to tabular data.

ACKNOWLEDGMENTS

Jayoung Kim and Chaejeong Lee equally contributed. Nuseong Park is the corresponding author. This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (10% from No. 2020-0-01361, Artificial Intelligence Graduate School Program at Yonsei University and 70% from No. 2021-0-00231, Development of Approximate DBMS Query Technology to Facilitate Fast Query Processing for Exploratory Data Analysis) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (20% from No. 2021R1F1A1063981).

REFERENCES

[1] Commonwealth of Australia 2010 Bureau of Meteorology. https://www.kaggle.com/ajayy/weather-dataset-rattle-package.
[2] HackerEarth Machine Learning Challenge—Adopt a buddy. https://www.hackerearth.com/akash14/adopt-a-buddy.
[3] Jonas Adler and Sebastian Lanz. 2018. Banach Wasserstein GAN. In NeuIPS.
[4] Martin Arjovsky, Soumith Chintala, and Leon Bottou. 2017. Wasserstein Generative Adversarial Networks. In ICML.
[5] Christopher M. Bishop. 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag.
[6] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. 1984. Classification and Regression Trees. Taylor & Francis. https://books.google.co.kr/books?id=JwQX-WOMs5yc.
[7] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority over-Sampling Technique. J. Artif. Int. Res.
[8] I. Giraudel and Sebastian Lanz. 2018. Banach Wasserstein GAN. In NeuIPS.
[9] Edward Choi, Siddharth Biswal, A. Bradley Maline, Jon Duke, F. Walter Stewart, Augustus Odema, Christopher Olah, and Jonathon Shlens. 2019. Generative Adversarial Minority Oversampling. In IJCV.
[10] Guang Zhao. 2006. Improved conditional GANs for tabular data generation. In IJCAI.
[11] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: A New over-Sampling Method in Imbalanced Data Sets Learning. In ICIC.
[12] Haibo He, Yang Bai, Eduardo A. Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In ICDM.
[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In NeuIPS.
[14] Justin Engelmann and S. Lessmann. 2021. Conditional Wasserstein GAN-based Tabular Data Generation with Recurrent Conditional GANs. In NeurIPS.
[15] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved Training of Wasserstein GANs. In NeuIPS.
[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved Training of Wasserstein GANs. In NeuIPS.
[17] Jinyoung Kim, Jinsung Yoon, and V. D. Mihaiela Schaar. 2019. PATE-GAN: Generalizing Synthetic Data with Differential Privacy Guarantees. In International Conference on Learning Representations.
[18] Jinyoung Kim, Jinsung Yoon, and V. D. Mihaiela Schaar. 2019. PATE-GAN: Generalizing Synthetic Data with Differential Privacy Guarantees. In International Conference on Learning Representations.
[19] Jinyoung Kim, Fenglong Xu, and Youngmin Kim. 2018. MMGAN: Manifold-Matching Generative Adversarial Networks. In ICLR.
[20] Ki Young Kim, Jinsung Yoon, and Youngmin Kim. 2018. MMGAN: Manifold-Matching Generative Adversarial Networks. In ICLR.
[21] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[22] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[23] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[24] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[25] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[26] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[27] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[28] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[29] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[30] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[31] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[32] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[33] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[34] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[35] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[36] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[37] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[38] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[39] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[40] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[41] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[42] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[43] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[44] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[45] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[46] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[47] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[48] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[49] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[50] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[51] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[52] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[53] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[54] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[55] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[56] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[57] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[58] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
[59] Kohei Tanaka and Shogo Takase. 2020. Adversarial minority oversampling for imbalanced tabular data. In IJCAI.
Table 7: The best hyperparameters of SOS we used in Table 2. \((\min, \max)\) means \((\beta_{\min}, \beta_{\max})\) for VP and Sub-VP or \((\sigma_{\min}, \sigma_{\max})\) for VE, respectively.

| Dataset       | SDE Type | Hyperparameters for SOS |
|---------------|----------|-------------------------|
|               |          | Layer Type (dim(h1), dim(hl), \ldots, dim(hdim)) | Activation | Learn. Rate (min, max) | Pred. | Corr. | SNR |
| Default       | VE       | Concat (512, 1024, 1024, 512) | LeakyReLU | 2e-03 | RD | Langevin 0.05 |
|               | VP       | (512, 1024, 1024, 512, 1024, 512) | LeakyReLU | 2e-03 | RD | Langevin 0.05 |
|               | Sub-VP   | (512, 1024, 1024, 512) | LeakyReLU | 2e-03 | RD | Langevin 0.05 |
| Shoppers      | VE       | Concat (512, 1024, 2048, 1024, 512) | ReLU | 1e-03 | RD | Langevin 0.05 |
|               | VP       | (512, 1024, 2048, 1024, 512) | ReLU | 2e-03 | RD | Langevin 0.05 |
|               | Sub-VP   | (512, 1024, 2048, 1024) | ReLU | 2e-03 | RD | Langevin 0.05 |
| Surgical      | VE       | Squash (512, 1024, 1024, 512) | ReLU | 2e-03 | RD | Langevin 0.05 |
|               | VP       | (512, 1024, 1024, 512) | ReLU | 2e-03 | RD | Langevin 0.05 |
|               | Sub-VP   | (512, 1024, 1024) | ReLU | 2e-03 | RD | Langevin 0.05 |
| WeatherAUS    | VE       | Concat (512, 1024, 2048, 1024, 512) | LeakyReLU | 2e-04 | RD | Langevin 0.16 |
|               | VP       | (512, 1024, 2048, 1024, 512) | LeakyReLU | 2e-04 | RD | Langevin 0.16 |
|               | Sub-VP   | (512, 1024, 2048) | LeakyReLU | 2e-04 | RD | Langevin 0.16 |
| Buddy         | VE       | Concat (512, 1024, 1024, 512) | SoftPlus | 2e-03 | PD | None |
|               | VP       | (512, 1024, 1024, 512) | SoftPlus | 2e-03 | PD | None |
|               | Sub-VP   | (512, 1024, 1024) | SoftPlus | 2e-03 | PD | None |
| Satimage      | VE       | Concat (512, 1024, 2048, 1024, 512) | LeakyReLU | 2e-04 | RD | Langevin 0.05 |
|               | VP       | (512, 1024, 2048, 1024, 512) | LeakyReLU | 2e-04 | RD | Langevin 0.05 |
|               | Sub-VP   | (512, 1024, 2048) | LeakyReLU | 2e-04 | RD | Langevin 0.05 |

Table 8: The best hyperparameters of the fine-tuning process we used in Table 2

| Dataset       | SDE Type | Hyperparameters for fine-tuning |
|---------------|----------|---------------------------------|
|               |          | \(\epsilon_f\) Learn. Rate \(\xi\) \(w\) Epoch Option |
| Default       | VE       | 5e-04 | 2e-06 | 80 | 0.90 | 4 | Reg. |
|               | VP       | 2    | 2e-08 | 80 | 0.95 | 1 | Bnd. |
|               | Sub-VP   | 5e-04 | 2e-06 | 90 | 0.95 | 1 | Bnd. |
| Shoppers      | VE       | 5e-04 | 2e-07 | 90 | 0.80 | 1 | Bnd. |
|               | VP       | 5e-04 | 2e-06 | 100 | 0.95 | 3 | Reg. |
|               | Sub-VP   | 5e-04 | 2e-05 | 80 | 0.95 | 2 | Bnd. |
| Surgical      | VE       | 5e-04 | 2e-08 | 80 | 0.99 | 3 | Reg. |
|               | VP       | 5e-04 | 2e-06 | 100 | 0.90 | 5 | Reg. |
|               | Sub-VP   | 5e-04 | 2e-07 | 80 | 0.95 | 3 | Bnd. |
| WeatherAUS    | VE       | 5e-04 | 2e-05 | 100 | 0.70 | 4 | Reg. |
|               | VP       | 5e-04 | 2e-07 | 60 | 0.60 | 1 | Reg. |
|               | Sub-VP   | 5e-04 | 2e-07 | 100 | 0.95 | 1 | Bnd. |
| Buddy         | VE       | 5e-04 | 2e-07 | 80 | 0.99 | 3 | Bnd. |
|               | VP       | 5e-04 | 2e-07 | 80 | 0.90 | 3 | Bnd. |
|               | Sub-VP   | 5e-04 | 2e-08 | 80 | 0.95 | 4 | Bnd. |
| Satimage      | VE       | 5e-04 | 2e-07 | 80 | 0.95 | 4 | Reg. |
|               | VP       | 5e-04 | 2e-06 | 80 | 0.70 | 5 | Reg. |

C SCORE NETWORK ARCHITECTURE

The proposed score network \(S_\theta(x, t)\) is as follows:

\[ h_0 = x_t, \]
\[ h_i = \omega(h_{i-1} \oplus h_{i-1}), 1 \leq i \leq d_N \]
\[ S_\theta(x, t) = FC(h_{d_N}). \]

where \(x_t\) is a record (or a row) at time \(t\) in tabular data and \(\omega\) is an activation function. \(d_N\) is the number of hidden layers. For various
layer types of $H_i(h_{i-1}, t)$, we provide the following options:

$$H_i(h_{i-1}, t) = \begin{cases} 
    FC_i(h_{i-1}) \odot \psi(FC'_i(t)), & \text{if Squash}, \\
    FC_i(t \oplus h_{i-1}), & \text{if Concat}, \\
    FC_i(h_{i-1}) \odot \psi(FC^\text{gate}_i(t) + FC^\text{bias}_i(t)), & \text{if ConcatSquash}, 
\end{cases}$$

where we can choose one of the three possible layer types as a hyperparameter, $\odot$ means the element-wise multiplication, $\oplus$ means the concatenation operator, $\psi$ is the Sigmoid function, and $FC$ is a fully connected layer.

### D SPACE AND TIME OVERHEADS

Some SGMs are notorious for their high computational overheads. In the case of tabular data, however, it is not the case since the number of columns is typically much smaller than other cases, e.g., the number of pixels of images. We introduce our training and synthesis overheads. As shown in Fig. 7, our training curves are smooth and in Table 10, the space and time overheads of our method are well summarized. The GPU memory requirements are around 4GB, which is small. It takes well less than a second per epoch for training. For the total generation time, i.e., the time taken to accomplish the entire oversampling task, our method shows less than 5 seconds in all cases. Sometimes, it takes less than a second.

### E FULL FAKE TABLE SYNTHESIS

Instead of oversampling minor classes, one can use our method for generating fake tabular data entirely — in other words, fake tabular data consists of only fake records. To this end, we train one score network of SOS (Sub-VP) with all classes without the fine-tuning process since we do not distinguish major/minor classes but try to generate all classes. In Table 9, we summarize its results in three datasets. As shown, it overwhelms all other existing methods. Even in comparison with Identity, it shows higher F1 scores in Shoppers and Satimage. However, its accuracy is lower than Identity in Shoppers and Surgical.

One interesting point is that for Satimage, our fake table shows better scores than Identity for all metrics. We compare the loss curves of the MLP classifier with the original data and our fake data in Fig. 8. Surprisingly, the fake tabular data by SOS yields lower training loss values than Identity. For some reasons, the fake tabular data is likely to provide better training, which is very promising. We think that it needs more study on synthesizing fake tabular data with SGMs to fully understand the phenomena.

---

**Table 9: Results of experiments synthesizing full fake tabular data**

| Method  | Shoppers | Surgical | Satimage |
|---------|----------|----------|----------|
|         | Acc.     | F1       | Acc.     | F1       | Acc.     | F1       |
| Identity| 0.883±0.002 | 0.503±0.005 | 0.827±0.002 | 0.608±0.005 | 0.846±0.002 | 0.799±0.003 |
| OCT-GAN | 0.778±0.009 | 0.440±0.012 | 0.716±0.004 | 0.472±0.003 | 0.735±0.010 | 0.699±0.007 |
| TVAE    | 0.835±0.011 | 0.490±0.010 | 0.695±0.026 | 0.527±0.014 | 0.798±0.008 | 0.767±0.009 |
| TableGAN| 0.854±0.024 | 0.515±0.013 | 0.752±0.003 | 0.496±0.009 | 0.783±0.007 | 0.710±0.007 |
| SOS     | 0.857±0.004 | 0.476±0.009 | 0.778±0.004 | 0.427±0.019 | 0.825±0.004 | 0.779±0.008 |

**Table 10: The runtime and GPU memory usage (averaged across classes) of SOS (Sub-VP) with the EM predictor.**

| Dataset   | Avg. Class size | GPU Memory Usage (MB) | Train Time per Epoch (sec) | Total Gen. Time (sec) |
|-----------|-----------------|-----------------------|---------------------------|------------------------|
| Satimage  | 3,186           | 4,424                 | 0.036                     | 4.240                  |
| Shoppers  | 6,165           | 4,608                 | 0.067                     | 0.824                  |
| Surgical  | 7,318           | 4,616                 | 0.185                     | 0.952                  |
| Buddy     | 9,417           | 4,357                 | 0.157                     | 1.718                  |
| Default   | 15,000          | 4,196                 | 0.395                     | 0.901                  |
| WeatherAUS| 28,210          | 4,812                 | 0.606                     | 1.766                  |