Applied Intelligence (2020) 50:341–353
https://doi.org/10.1007/s10489-019-01515-3

Using deep learning to preserve data confidentiality

Wei Li[1] · Pengqiu Meng[2] · Yi Hong[3] · Xiaohui Cui[1]

Published online: 24 July 2019

Abstract
Preserving data confidentiality is crucial when releasing microdata for public-use. There are a variety of proposed approaches; many of them are based on traditional probability theory and statistics. These approaches mainly focus on masking the original data. In practice, these masking techniques, despite covering part of the data, risk leaving sensitive data open to release. In this paper, we approach this problem using a deep learning-based generative model which generates simulation data to mask the original data. Generating simulation data that holds the same statistical characteristics as the raw data becomes the key idea and also the main challenge in this study. In particular, we explore the statistical similarities between the raw data and the generated data, given that the generated data and raw data are not obviously distinguishable. Two statistical evaluation metrics, Absolute Relative Residual Values and Hellinger Distance, are the evaluation methods we have decided upon to evaluate our results. We also conduct extensive experiments to validate our idea with two real-world datasets: the Census Dataset and the Environmental Dataset.

Keywords Data confidentiality · Generative model · Statistical evaluation metric

1 Introduction

When data collectors release microdata to analyzers, the issue of honoring confidentiality constraints regarding the data has always been important. Many researchers focus on this challenge and they have developed many techniques [1] [2] [3] [4] to deal with this issue at hand. Developed techniques include Statistical Disclosure Limitation (SDL) including topcoding or bottomcoding [5], data swapping [6], adding random noise [7], and multiple imputation [1]. Nonetheless, there remain fundamental issues in the developed techniques. One of the typical issues in these techniques is that the released data cannot completely be masked or replaced, so the risk of confidentiality exposure still exists. One technique, multiple imputation, is shown in Fig. 1 sub-figure (a). The two variables *X1* and *X2* are shown in the left table of the sub-figure (a). The multiple imputation technique generates synthetic data via posterior predictive distribution (e.g., *P (X2 | X1)* or *P (X1 | X2)*). This synthetic data is denoted with red and blue filled circles in the right table of sub-figure (a). From the sub-figure (a), we can see that the multiple imputation technique [1] cannot mask all data. In general, people denote an $n \times p$ data indicator matrix $Z = z_{i,j}, ..., z_{n,p}$ where $z_{i,j} = 1$ indicates that $x_{i,j}$ is selected to be replaced by synthetic value while $z_{i,j} = 0$ indicates that $x_{i,j}$ is unchanged, given that the original dataset has been set as $X = x_{i,j.}$ The issue rises in the fact that unchanged data could be sensitive, resulting in a risk of exposure of confidentiality.

As for the bottomcoding technology shown in sub-figure (b), it roughly sets a threshold (here threshold = 4) to replace data with values less than or equal to the threshold. This presents two challenges: 1) some sensitive data above the threshold is still exposed; 2) the intruder could infer the sensitive data from non-sensitive data. Considering the potential challenges of these approaches, there is room for a new approach that eliminates these issues.

✉ Xiaohui Cui
xcui@whu.edu.cn

Wei Li
auto_weili@whu.edu.cn

Pengqiu Meng
mengp@wustl.edu

Yi Hong
yihongucla@gmail.com

[1] School of Cyber Science and Engineering, Wuhan University, Wuhan City 430079, China

[2] School of Engineering & Applied Science, Washington University, St. Louis, MO 63130, USA

[3] Sumo Logic Inc., Redwood City, CA, USA

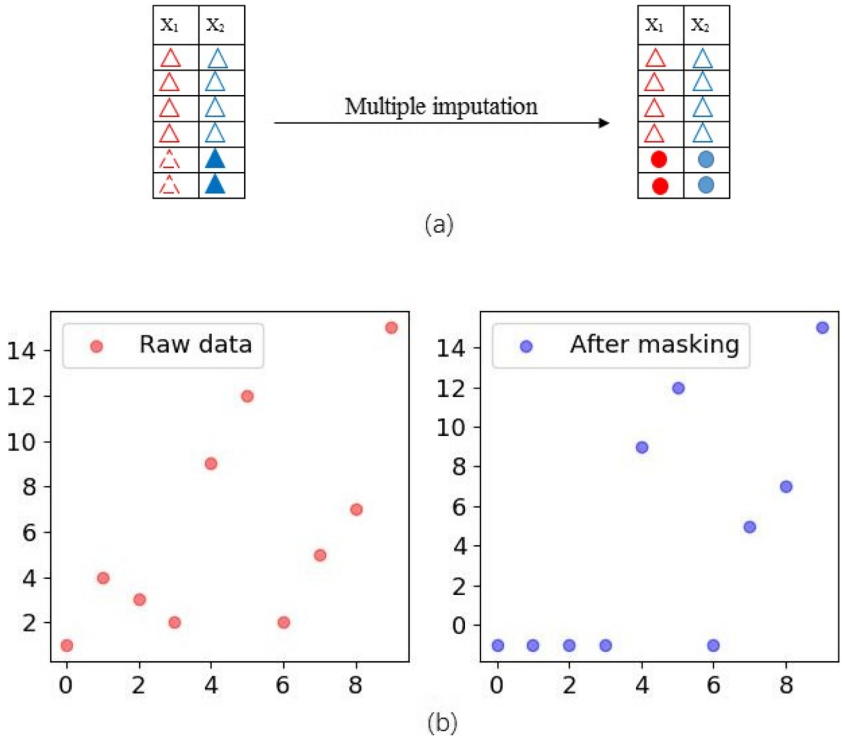


Fig. 1 Sub-figure (**a**) demonstrates an example in which we apply the multiple imputation technology [1] to the original data. The *X1* and *X2* jointly denote the original data, and the red dash-line and blue-filling triangles represent the sensitive data that we want to mask at the left table. In the right table, the red-filling and blue-filling circles are synthetic data, which means that these sensitive data have been masked by the multiple imputation technique. Sub-figure (**b**) shows a scenario where we apply the bottomcoding technology [5] to a dataset {1,4,3,2,9,12,2,5,7,15}. We set a threshold (<=4) to replace any number that is less than or equal to 4 with −1. The results show that although original data is partially masked, it still leaves rooms for confidentiality exposure

In this study, instead of utilizing mathematical statistics to mask or replace raw data, we explore an alternative masking direction, which is to use the deep learning-based generative model to generate simulation data to replace original data that consists of similar statistical characteristics (each bin has the similar/same statistical frequency shown in Fig. 2) to the original data. For the state-of-the-art generative models (AutoEncoder [9] and Generative Adversarial Net (GAN) [10]), the mechanism of generating simulation data is to transform a noise, sampled from an easy-to-sample distribution (e.g., Uniform distribution or Gaussian distribution), into the "realistic" data. The training would be complete when the discriminator is unable to differentiate between the raw data and the generated data. Even when this occurs, we need to investigate if similar distribution means similar statistical characteristics. To visualize our hypothesis, we test the relationship between statistical characteristics and similar distributions. This is shown in Fig. 2.

In Fig. 2, the sub-figures (a) and (b) correspond to statistical results of original data and simulation data, respectively. Subfigure (c) royally reflects both the original data Gaussian distribution and the generated data Gaussian distribution. As is readily apparent, the age results shown in sub-figure (a) are significantly different from (b). The age range of the former histogram (a) is 0 to 90, while that of the latter histogram (b) is limited from 20 to 45. Although both Gaussian distributions shown in sub-figure (c) are similar, the generated data cannot represent the original data in practice. In other words, they are two totally different datasets. Also, the generated data is concentrated on a limited interval, which implicitly indicates a lack of diversity.

A few elements need to be considered for creation of the generative model. To generate simulation data consistent with the original statistical data, we first have to test which generative model (AutoEncoder or GAN) is more suitable for generating numeric data (this will be discussed in Section 3). We, also, consider the optimizer impact on the quality of generated data by the generative model. Different optimizers influence the training stability which can adversely affect the generated data quality (Adam even turns the gradients negative in Wassertein GAN [11]). Additionally, evaluation methods need to be discussed. Generative models usually rely on the two distributions approaching each other. However, Fig. 2 shows that the two approaching distributions cannot guarantee both raw data and generated data have the same statistical frequency in the histogram. We utilize two evaluation metrics, Absolute Relative Residual Values [12] and Hellinger Distance [13], to evaluate such statistical characteristics after mapping both simulated data and original data into the histogram. More details are shown in Section 3.

In summary, our major contributions are as follows:

- This paper explores which deep learning-based generative model (AutoEncoder or GAN) is suitable for generating the numerical dataset.
- This paper casts light on adding a batch normalization function into the GAN mode's hidden layer and choosing the SGD optimizer to best modify the model. In particular, this paper demonstrates how to utilize batch normalization to generate numerical simulation data and this paper discusses the optimizer impact on the quality of generated data.
- We propose two evaluation metrics to assess the similarity between the generated data and the raw data, after mapping them into the histogram.
- Through comprehensive experiments on two datasets, we demonstrate the effectiveness of the proposed approach.

This paper is organized as follows. In Section 2 we discuss some related work. In Section 3, we present our main idea and evaluation metrics. In Section 4 we show our experiment results. Section 5 serves as the conclusion.

2 Related work

Since directly releasing micro-data brings the risk of privacy exposure, the method Statistical Disclosure Limitation (SDL) [14] has been developed for protecting the confidentiality of released data. The topcoding method [5] replaces the data samples (x) with a threshold C when $x \geq C$. Similarly, the bottomcoding method replaces data samples (x) when $x \leq C$. Adopting this similar strategy (using a threshold to synthesize data) also includes t-closeness [15]. Researchers use recording method [5] to edit raw data when releasing it categorically. For example, they hide specific age values in specified intervals in a histogram. Challenges exist with this method: while it narrows the data range, malicious analysts can still discover the original data. Another data swapping technique is based on the idea of swapping sensitive data with non-sensitive data to reduce the risk of exposure. This technique is optimal for preserving univariate distribution while unsatisfying multivariate relations in a dataset. The scenario where skilled, malicious analysts can use the non-sensitive data to infer sensitive data, will inevitably, still exist. Adding random noises [7] can confuse the intruders. However, such a method cannot satisfy the covariance structure of the raw data, and too much noise can also confuse benevolent analyzers. In summary, we can see that raw data cannot be completely covered with SDL, because the risk of leaky confidentiality still exists. An alternative approach to SDL is multiply-imputed synthetic micro-data [1]. A multiply-imputed dataset consists of the actual data (X), the designer (D), the outcome variables with some confidentiality (Y) and the outcome variables without confidentiality (Z). The challenge is how to predict (Z, Y) from X with D. Note that such a technique can work in particular cases and not all scenarios.

A recently active area of research is to use the differential privacy [16, 17] to preserve the data privacy. In the traditional process of retrieve, for example, we assume an intruder knows the expected information in the 3rd row (all information is listed row by row), and he does not know the private, specific attribute value at the same place. The intruder can, however, obtain the private value by using a *SQL* sentence, i.e., *count(3)-count(2)*. Differential privacy injects some noise or disturbance data to protect the raw data. Specifically, a disturbance can be utilized on a given dataset A to render a disturbed dataset A'. After that, we randomly delete a row from A and we perform the same strategy; we name the new dataset B'. If B' is similar to A', we think the differential method worked. Note that such a method does not concern whether A' is similar to A. The disturbed dataset A' may hold different statistical

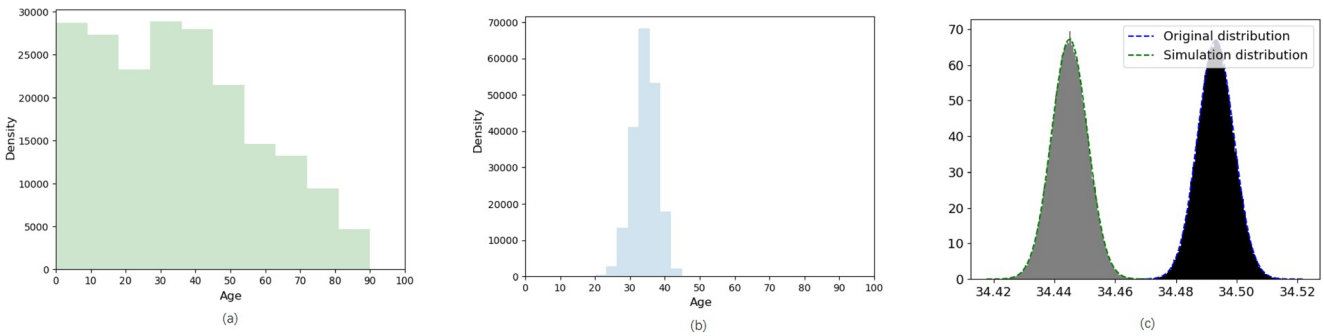


Fig. 2 Sub-figure (**a**) indicates the histogram of raw data of age field in census dataset, while sub-figure (**b**) indicates the histogram of generated data produced by a regular GAN model with Adam gradient [8] on the same field. In comparison with raw data histogram, here we produce generated data with same size as raw data. Since the age-bracket is from 0 to 90, we map all raw or generated values into the range of (0, 90), and we use an array of size 100 to save the generated values. We count the statistical percentile of each entry in this array. After that, we use the *plt.hist()* function with bins = 10 to plot their histograms. Sub-figure (**c**) shows the Gaussian distributions of both original data and generated data. From the three sub-figures, we can see that although the two distributions are closely similar, the generated data cannot represent the original data in practice. This is because they have different statistical characteristics

characteristics than the raw dataset *A*. More details are shown in the experiments section.

In this study, our idea is to use simulation data to address the aforementioned challenges. Simulation data is already widely utilized and has been applied to many fields including agriculture [18], astronomy [19], visualization [20], crime analysis [21], biology [22] and transportation [23]. These generation techniques are often domain-specific or problem-specific. One recent generative model is Generative Adversarial Net (GAN) [10]. It can generate quite different simulation data when compared with the raw data (e.g.," *King"* -" *Man"* +" *Woman"* = *"Queen"*) [24], which can completely cover the original data. The GAN model consists of two components, a discriminator (*D*) and a generator (*G*). The discriminator (*D*) can be viewed as a detective who can tell whether a data point is genuine or not, and the generator (*G*) can be regarded as a forger who generates simulation data and tries to fool the discriminator into accepting generated data as real. If the discriminator cannot distinguish whether a sample is genuine or not (e.g., $D(G(z)) = 0.5$), the training process is complete. The potential of GAN for generating simulation data has been an area of active research, and its extensions, such as Conditional GAN [25], Deep Convolutional GAN [24], Wassertein GAN [11] and Least Squares GAN [26], have been widely adopted.

Another generative model, the AutoEncoder [9], also consists of two components, an Encoder and a Decoder. It adopts Compression-Reconstruction technology to generate simulation data. In the compression stage, the Encoder would compress input samples to hidden nodes; while in the reconstruction step, the Decoder would reconstruct samples from the hidden nodes. There is an equivalence relationship between input and output (e.g., $X_{\text{input}} \approx X_{\text{output}}$). If the model achieves such equilibrium, we can feed noise (the size of the noise equal to that of the hidden nodes) into the Decoder to synthesize new instances. Dai et al. [27] attempt to use the Encoder-Decoder to protect private information. They use this model to extract the privacy region in videos and scramble it while encoding so the users can only see the non-private regions, given that the AutoEncoder can be used to reduce dimensionality. Psychoula et al. [28] also uses the Encoder-Decoder system to preserve sensitive personal data, based on the user-access authority.

These two generative models mainly focus on image generation and classification. The application to numeric data has been very limited. Also, while there are many evaluation metrics for generative model, few evaluation metrics focus on whether the simulated data has the same statistical characteristics as the original data. In this study, we extend the application of deep generative models from image generation to generic numeric data generation. We also introduce two statistical metrics to assess the statistical similarity between the generated data and the original data.

3 Simulation data generation and statistical gap measure

In this section, three issues regarding simulation data generated from deep learning-based generative model will be discussed: 1) which generative model is suitable for numerical data generation? 2) a new generating process. 3) how to assess the statistical similarity between generated data and original data?

3.1 Performance assessment of generative model

Although the two generative models were introduced in Section 2, we formally describe them below to establish continuity. The goal of AutoEncoder is to learn representations (encoding) from a dataset, especially for the purpose of dimensionality reduction. Recently, this model has become widely used for generating simulation data. It always consists of two components, the encoder and the decoder, which can be defined as transitions φ and ψ respectively:

$$\varphi, \Psi = \text{argmin}_{(\varphi\ \Psi)} \|X - (\varphi^{\circ}\Psi)\cdot X\|^2 \tag{1}$$

In which, $\varphi: \chi \to F$ and $\psi: F \to \chi$. There is one hidden layer in Eq.(1), the encoder takes the input $x \in R_{\text{d}} = \chi$ and maps it to $z \in R_{\text{p}} = F$. Since the best generative model for AutoEncoder is the Adversarial AutoEncoder (AAE) [29], we adopt the AAE to generate simulation data. The AAE is regularized by matching the aggregated posterior, *q(z)*, to an arbitrary prior, *p(z)*. The encoder ensures the aggregated posterior probability distribution can fool the discriminator into accepting the hidden code *q(z)* that comes from the prior probability distribution *p(z)*.

A GAN model usually consists of two components [10], a generative model *G* and a discriminative model *D*. The generator *G* transforms an input, which is a latent random vector *z* sampled from noise distribution $p_z(z)$, into an image, and then the image is fed into the discriminator *D*, given that the training dataset is an image dataset. The discriminator *D* would output a single scalar to indicate that the current data come from the generator rather than original dataset. Thus, the two components are competing against each other.

We simultaneously train the generator and the discriminator by using the following loss Eq. (2):

$$\begin{aligned} minmax_{(G,D)} V(G,D) = {} & Ex{\sim}p_{data(x)} logD(x) \\ & + E_{z\sim pz}(z)[log(1-D(G(z)))] \end{aligned} \tag{2}$$

In Eq. (2), *x* comes from a distribution $p_{\text{data}}(x)$ sampled from the original dataset and z comes from another distribution $p_z(z)$ sampled from the noise. The generator would produce the simulation data to fool the discriminator into accepting it, and the discriminator would output a single score

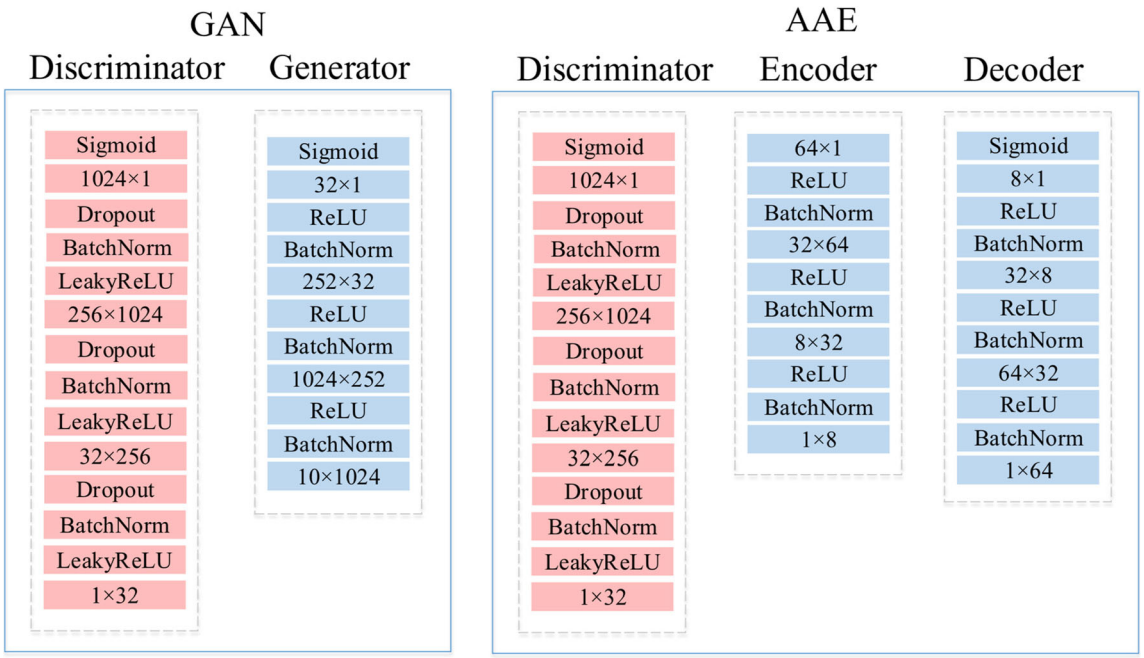


Fig. 3 Architectural details of AAE and GAN. The weights are set Normal (0.0, 0.02) and the biases are set (0.0). The learning rate is 0.0002. We use *BCE* to update both generator's parameters and discriminator's parameters. For discriminator, we set the *LeakyRelu* [31] as 0.02 and that of *Dropout* [32] as 0.5. Two models apply BatchNorm1d to each hidden layer and the optimizer adopts SGD

$\in [0, 1]$ to indicate whether the evaluated data is from the generator or not. Generally, a smaller score indicates that the current data is from the generator, while a higher score indicates that the current data comes from the original dataset. We repeat this training process until both the discriminator and the generator reach a Nash Equilibrium [30] where $p_G(z) = p_{data}(x) = 0.5$.

We then explore the first issue: which generative model is more suitable for generating numeric simulation data. We utilize similar hyper parameters to fairly compare the two models (regular AAE and regular GAN). Specifically, the implementation details of the two models are shown in Fig. 3. The generated results are shown in Fig. 4.

In Fig. 4, the sub-figure (a) indicates the generator loss and that of decoder loss while sub-figure (b) reflects the two models' histograms. From the results, we can see that the AAE model is not suitable for our case. Not only is the loss divergent but the statistical percentile of generated data is different from the raw data (as is shown in sub-figure (a) of Fig. 2). This is because the AutoEncoder model spreads probability mass to places it might not make sense [33]. In addition, the AutoEncoder model belongs to lossy compression

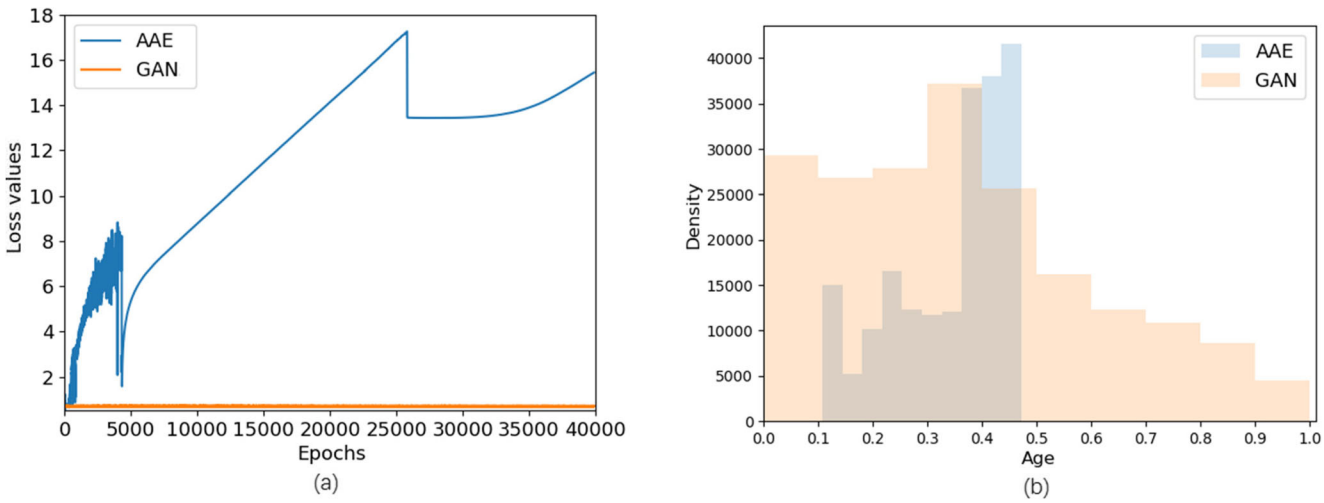


Fig. 4 The two generative models have been applied on age field in census dataset. Apparently, the performance of GAN is better than that of AAE. Sub-figure (**a**) indicates that the loss for decoder of AAE cannot converge while that is stable for generator of GAN, and (**b**) indicates the corresponding statistical results of two generations from both AAE and GAN

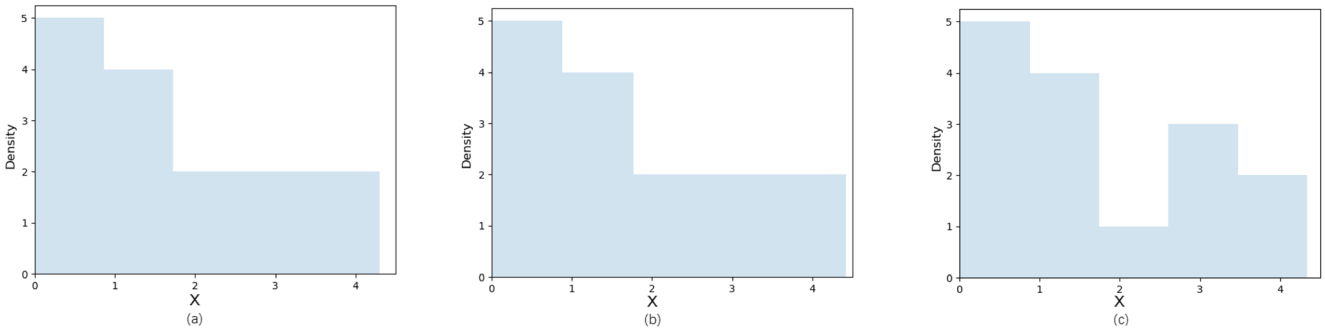


Fig. 5 An example to demonstrate histogram-based similarity measure. Sub-figure (**a**) indicates the original statistical results, while sub-figures (**b**) and (**c**) indicate the different statistical results of generated data

community. Much of the original information is inevitably lost in the compression layer of the Encoder. This is inherent to the process of dimensionality reduction and can cause an imperfect reconstruction [34]. If the decoder cannot fully learn the salient features of raw data, it also cannot transform noise z into realistic-like data using the interpolation method. The performance of the AutoEncoder model cannot be guaranteed under such a scenario. Therefore, we deem the GAN model to be more suitable for our task than AutoEncoder.

3.2 Simulation data generation with GAN

Although the GAN model has been widely used in many fields, its application to numeric data generation has been very limited. While applying the GAN to the numeric dataset to generate simulation data is a trivial task, retaining the same, or similar, statistical characteristics as the raw data presents more of a challenge.

In Fig. 2, the data-span of raw data in sub-figure (a) is large whereas the histogram of generated data (sub-figure (b)) focuses on a narrow range $\in (30, 50)$. This indicates that the GAN model basically learns features belonging to samples within the range $\in (30, 50)$. In other words, such generated data loses the diversity of the original dataset. To be noted is the range $\in (30, 50)$ of raw samples holds a greater quantity of data than other intervals.

To learn the characteristics from all intervals, we scale the data range. In doing so the method can ignore a measure among different features while retaining the same distribution as the raw dataset. Additionally, since the GAN model can more easily product generation with an easy-to-sample distribution the generator more easily produces a distribution with a small data-span (e.g., Uniform distribution $\in [-1, 1]$ or Gaussian distribution $\in [0, -1]$).

The goal of the GAN model is to transform a distribution into another one. It holds no regard for retaining statistical characteristics of the raw data. In this study, for generating simulation data with statistical characteristics the same as the raw data, we add the batch normalization function [35] to each hidden layer, for both the discriminator and the generator, to re-scale the output from each hidden layer. Just like the activation function or the convolutional layer, the batch normalization is viewed as a layer of the network. In the low-layer neural network, the parameters are updated during training, changing the distribution of input in the next layer (this scenario is named 'Internal Covariate Shift'). The batch normalization can address this issue.

Moreover, when we compare Figs. 2 and 4, we found that the same model (GAN) uses different optimizers so the histogram displays different statistical characteristics. The performance shown in Fig. 4 is better than that of Fig. 2, indicating the optimizer is an important factor for generating numerical simulation data.

In general, the gradient method is closely related to an algorithm's convergence. If a gradient method is suitable for an algorithm, the algorithm would converge at the global optimum value. If it is not suitable for the algorithm, it would converge at the local optimum value. If the latter case scenario occurs, the model would not hold good generalizability and would perform poorly. There is not, however, consensus over which gradient method a network should choose. Recent works have had conflicting results. Kingma et al. [8] asserts the performance of Adam is better than SGD with faster convergence and avoidance of frequent fluctuation. Other studies hold the opposite view. Hardt et al. [36] has shown that the SGD is uniformly stable, generalizes well and solutions with low training error are found quickly. Wilson et al. [37] observed that the solutions found by adaptive methods (like Adam, RMSProp) generalize worse (often significantly worse) than SGD; this study encourages practitioners to use

Table 1 Using the Absolute Relative Residual and the Hellinger Distance to calculate the similarity between two datasets. The Hellinger Distance is from the SCIPY

ID	Absolute relative residual	Hellinger distance
P(X, Y)	9.9114	0.0599
P(X, Z)	15.3235	0.134

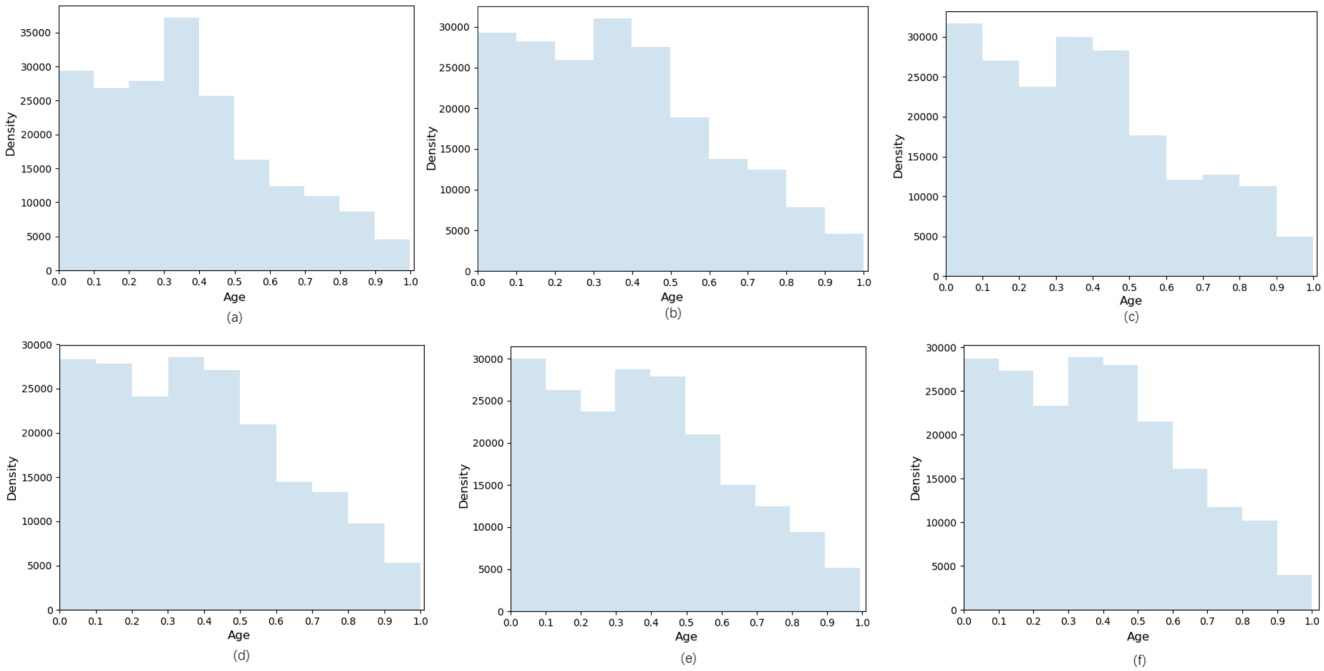


Fig. 6 Sub-figure (**a**) uses the Adam gradient method, and sub-figure (**b**) uses the Adagrad gradient method, and sub-figure (**c**) uses the RMSProp gradient method. Sub-figure (**d**) uses the SGD + Nesterov momentum, while sub-figure (**e**) only uses the SGD gradient method and (f) reflects the statistical results of age field data samples

the standard SGD method. WGAN [11] has also proven that the Adam optimizer is worse than RMSProp and SGD, because Adam may even turn the gradients negative. Since the adopted optimizer influences model performance and argument still exists over which optimizer performs better, in our study we adopt different optmizers in our generation of simulation data from the raw dataset. The details of this can be observed in the experiments section.

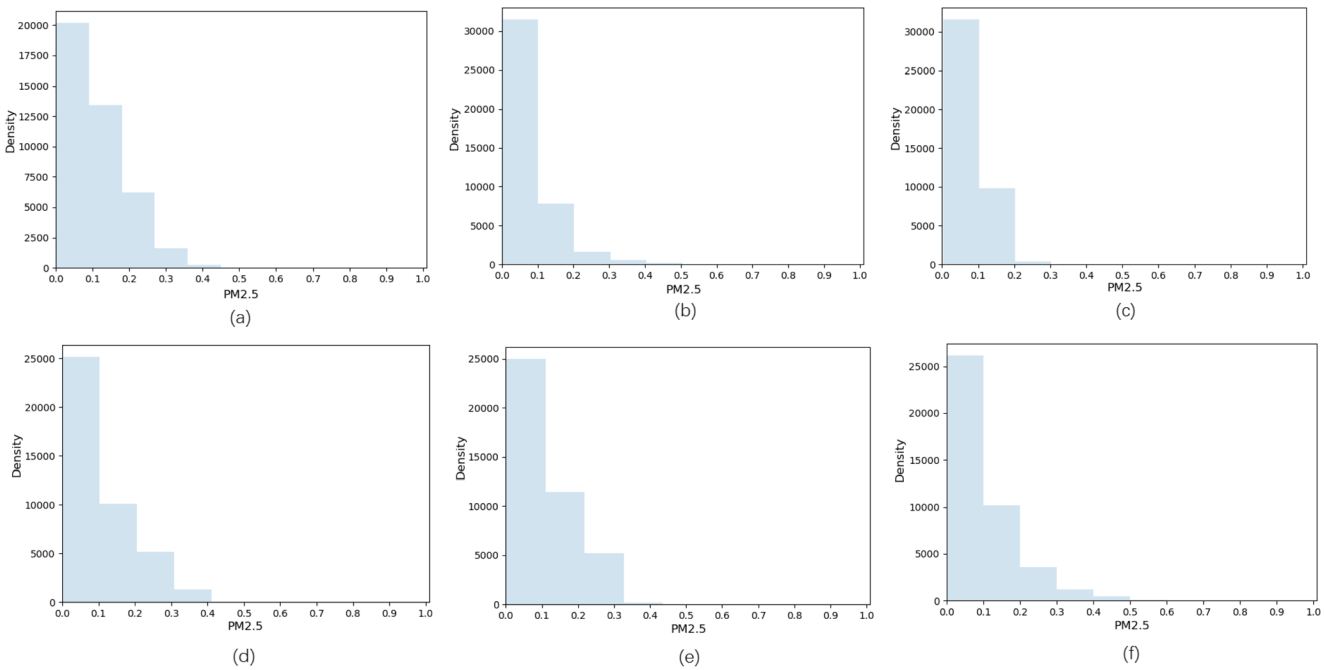


Fig. 7 Sub-figure (**a**) uses the Adam gradient method, and sub-figure (**b**) uses the Adagrad gradient method, and sub-figure (**c**) uses the RMSProp gradient method. Sub-figure (**d**) uses the SGD + Nesterov momentum, while sub-figure (**e**) only uses the SGD gradient method and (f) reflects the statistical results of PM2.5 data samples

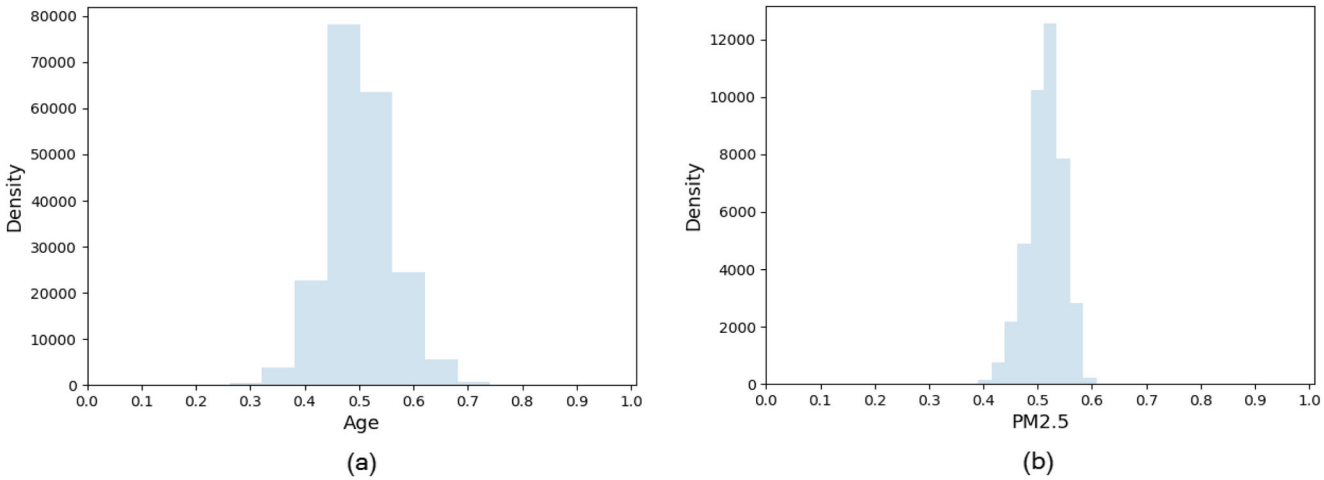


Fig. 8 Sub-figure (**a**) indicates that we apply the regular WGAN model to the age field while sub-figure (**b**) indicates that the regular WGAN model has been applied to the PM2.5 field

3.3 Evaluation metrics

In this subsection, we will discuss the statistical evaluation metrics. Let us refer back to sub-figure (a) in Fig. 2 and sub-figure (b) in Fig. 4, in which different simulation data has different statistical results. It remains an open question which one is better. We use the statistical gap in the histogram, a commonly used strategy [38], to measure the similarity between two datasets. We proceed to discuss this further.

Suppose there is a dataset X containing 15 samples [0.01, 0.2, 0.3, 0.4, 0.5,1.2, 1.1, 1.3, 1.6, 2.2, 2.5, 2.8, 3.3, 3.7, 4.3]. Sub-figure (a) in Fig. 5 represents this data. Sub-figure (b) and sub-figure (c) are the statistical results of different generated datasets Y and Z, which are [0.1, 0.25, 0.32, 0.44, 0.49, 1.15, 1.23, 1.3, 1.8, 2.35, 2.65, 2.75, 3.35, 3.68, 4.2] and [0.13, 0.14, 0.16, 0.178, 0.189, 1.2, 1.34, 1.56, 1.65, 2.55, 3.2, 3.35, 3.78, 4.21, 4.25], respectively. From the perspective of confidentiality, the two sub-figures (b) and (c) have masked the original data shown in sub-figure (a); from the perspective of statistics, the performance of sub-figure (b) is obviously better than that of sub-figure (c). Based on this example, the idea is to use statistical methods to output a specific score to evaluate our generation:

- Absolute Relative Residual Values (ARR) [12].
- Hellinger Distance (HD) [13].

The residual values [12] equation can calculate the relative difference between original data and generated data on each sub-interval (we view each bin as the sub-interval). The smaller the residual values are, the better the performance is. The equation is defined as follows:

$$V = \sum_{i=1}^{n} \left| \frac{hist_{xi} - hist_{zi}}{hist_{xi}} \right| \tag{3}$$

In Eq. 3, $hist_{xi}$ indicates the original statistical quantity at i-th sub-interval and $hist_{zi}$ corresponds to the statistical quantity of generation at i-th sub-interval.

In statistics, the Hellinger distance [13] is used to quantify the similarity between two probabilistic distributions, and it belongs to the f-divergence [39]. In our study, we use the probability percentile to quantify the similarity between two datasets. Assuming there are two discrete probability distributions $P = (p_1, p_2, ...p_n)$ and $Q = (q_1, q_2, ..., q_n)$, their equation is defined as follows:

$$H(P,Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} \left(\sqrt{p_i} - \sqrt{q_i}\right)^2} \tag{4}$$

We can reformulate Eq. (4) as the Euclidean distance of two square root vectors, which is shown as follows.

Table 2 The absolute relative residual values and Hellinger distances on age field with different gradient methods

Optimizer	Absolute Relative Residual	Hellinger distance
Adam	107.7575	0.059
Adagrad	32.4578	0.028
RMSProp	21.4728	0.035
SGD + Nesterov	12.6631	0.0103
SGD	**5.5679**	**0.009**

Table 3 The absolute relative residual values and Hellinger distances on income field with different gradient methods

Optimizer	Absolute Relative Residual	Hellinger distance
Adam	120.7629	0.275
Adagrad	42.4338	0.206
RMSProp	132.6314	0.341
SGD + Nesterov	31.8296	0.167
SGD	**17.1973**	**0.134**

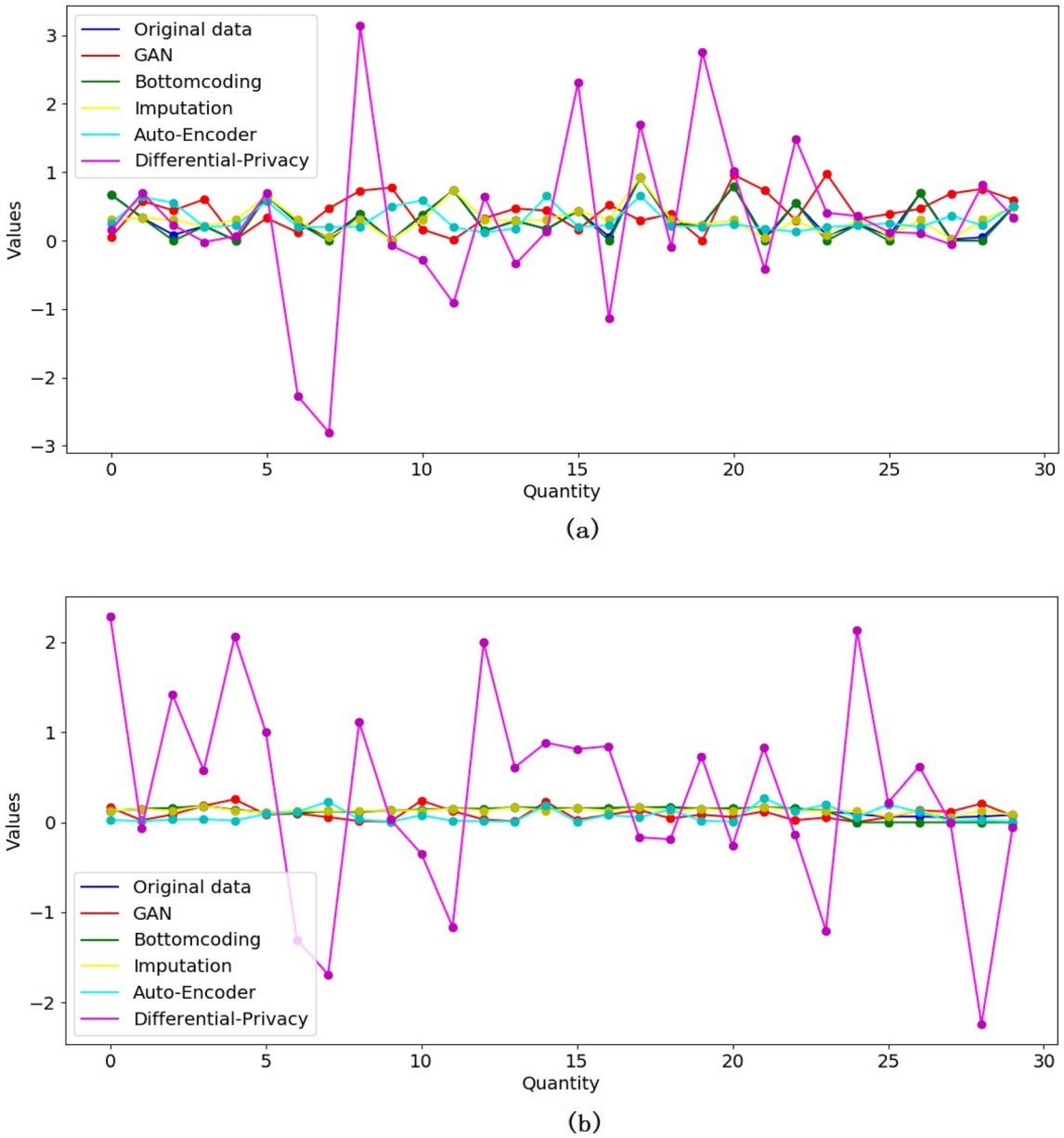


Fig. 9 We use the GAN model and four other data confidentiality methods (bottomcoding method, multiple imputation method, differential privacy method and Encoder-Decoder model) to mask the census dataset (sub-figure (**a**)) and the PM2.5 dataset (sub-figure (**b**)). We pick 30 samples for conveniently observing their differences. From the two results, we can see that the GAN model, the Encoder-Decoder model and the differential privacy method can mask all original samples. The other methods are partially masking the original samples. Notice that the Inception method is from the SKLEARN, and the Laplace function within the differential privacy method is from the NUMPY. As for the bottomcoding, the threshold is 0.2 for census dataset and 0.1 for PM2.5 dataset. Moreover, each circle node indicates a value

$$H(P,Q) = \frac{1}{\sqrt{2}}\left|\sqrt{P}-\sqrt{Q}\right| \tag{5}$$

Our idea is based on relative frequency, counting the number of each entry falling in specific sub-intervals. Assuming the samples are mapped into the interval [0,1], the two relative frequencies would be equivalent if the two datasets have the same probability distributions at each sub-interval. Thus, this measure can exactly evaluate the difference of statistical results on two datasets. The smaller the score is, the better the performance is. The two evaluation metrics are complementary to each other, we then use them to calculate the similarity between the dataset X and the dataset Y or Z. The results are shown in Table 1.

Figure 5 shows that the dataset Y is more similar to dataset X than dataset Z. The Absolute Relative Residual and the Hellinger Distance, shown for the values in Table 1, are reflective of this.

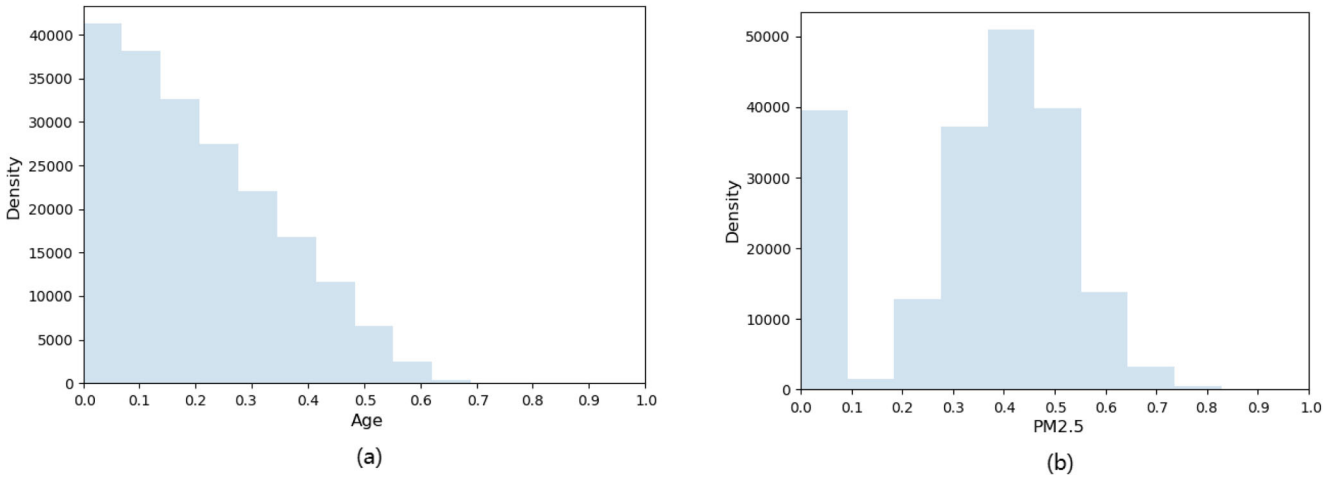


Fig. 10 We apply the Auto-Encoder model to the two datasets. Sub-figure (**a**) indicates that the Auto-Encoder has been applied to the age field in Census dataset, while sub-figure (**b**) indicates that it has been applied to the PM2.5 field in environmental dataset

4 Experiment

To validate our approach, we choose a census dataset extracted from the 1994 and 1995 population surveys conducted by the U.S. Census Bureau, and an environmental dataset containing the PM2.5 data of the US Embassy in Beijing between Jan 1st, 2010 to Dec 31st, 2014. Supposing the two datasets consist of sensitive information, our work is to use a generative model to generate simulation data to mask the original samples. We use the regular GAN model with batch normalization functions to generate the simulation data and compare the performance of different gradient methods (Adam [8], SGD [40], RMSProp [41], Adagrad [42] and Nesterov [43]). To validate the merit of our idea on data confidentiality, we would compare the GAN model with traditional data preservation approaches.

4.1 Data description

The census dataset includes 199,523 samples with 42 features. Since this dataset is a questionnaire, it contains many nature language answers and some missing values on some fields. We take 199,500 elements from the age field of the census dataset to validate our idea.

The environmental dataset includes 43,824 samples with 13 features. This dataset is also a questionnaire but it contains many missing values on some fields. We take 41,700 examples from the PM2.5 field of the environmental dataset as training data.

4.2 Simulation data generation

First, we generate data on the census dataset. The implementation details of the GAN model are shown in Fig. 3. We conduct experiments on age features with various optimizers. The corresponding results are shown in Fig. 6 (here we try with different optimizers while other hyper parameters remain constant). Figure 6 shows that the SGD function outperforms other optimizers.

From Fig. 6, the results shown in (d) and (e) are most similar to original histogram, which means that the SGD is better than other gradient methods when applied to numeric

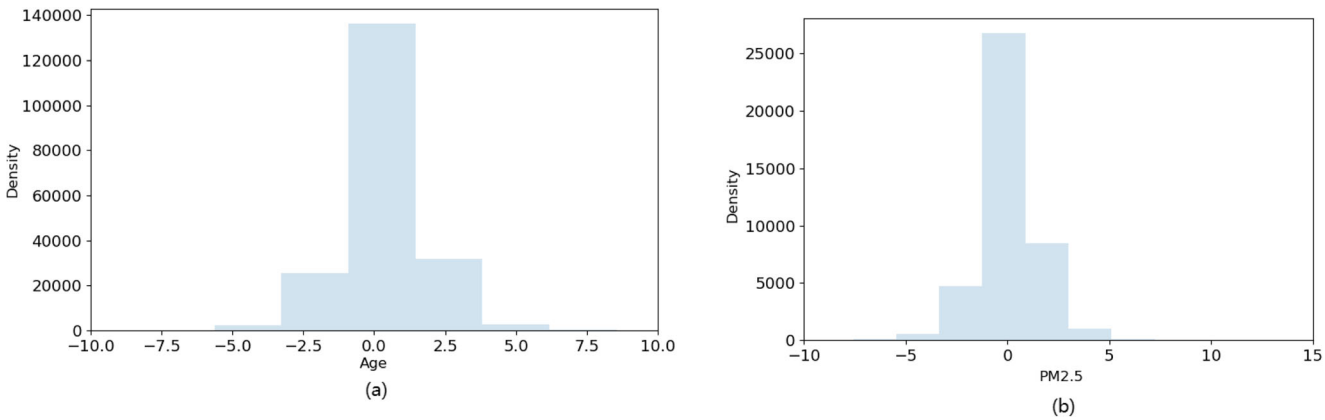


Fig. 11 We apply the differential privacy method to two datasets. Sub-figure (**a**) indicates that the differential privacy method has been applied to the age field in Census dataset, while sub-figure (**b**) indicates that it has been applied to the PM2.5 field in environmental dataset

data. We continue to apply this idea to the environmental PM2.5 dataset with the same hyperparameters, changing the batch size to 300. The results are shown in Fig. 7.

In this case, the performance of SGD gradient method still is better than other optimizers. There are variants of the basic GAN model [10] to choose from (described earlier). Here we adopt the WGAN model. The WGAN transforms the discriminative problem into a regression problem and the Wasserstein distance is a better metric than traditional Jensen-Shannon divergence, which the other GAN variants adopt. Other GAN variants have a similar generating mechanism as the regular GAN model. Figure 8 shows the generated results on Census and PM2.5 datasets with the WGAN model.

From Fig. 8, we can see the generated data performed poorly based on the differences in these histograms when compared with the raw data histograms (See sub-figures (f) of Figs. 6 and 7). We think that the unsatisfactory generated results are related to the weight clipping. For calculating the Wasserstein distance, WGAN utilizes the network with parameter θ, limited to a certain range (e.g., $\theta \in [-0.01, 0.01]$), to approach the Wasserstein distance. However, if the parameters of the network have been limited to a certain range, the outputs of the generator would also be limited to a certain range [44]. Based on these results, we recommend our idea as a promising data masking alternative to generate suitable numerical samples. We now further explore if the statistical characteristics of the generated data well mimics those of the original data.

4.3 Evaluation using statistical knowledge

In this work we focus on simulation data assessment. We measure the statistical similarity between generated data and original data by using Absolute Relative Residual Values and Hellinger distance. We first apply the two evaluation metrics (the Absolute Relative Residual and the Hellinger distance) to the age generation. The results are shown in Table 2.

From Table 2, we can see that the best performance is with the SGD optimizer. Both shape and statistics show that the simulation data generated by the GAN model with the SGD optimizer can royally reflect the statistical characteristics of raw data. We continue to apply the same evaluation metric to the PM2.5 field. The results are shown in Table 3.

From Table 3, we can see that the SGD is still the best. The two groups of results show that the SGD gradient method is better than other optimizer functions in the numeric dataset.

4.4 Comparing the GAN model with other data confidentiality methods

We then compare the GAN model with other data confidentiality methods (e.g., bottomcoding method [5], multiple imputation method [1], differential privacy method [16] and Encoder-Decoder model [29]) on the census dataset and PM2.5 dataset. The results are shown in Fig. 9.

In Fig. 9, we can see that the GAN model [10], the Encoder-Decoder model [29] and the differential privacy method [16] can mask all original samples. Each value (circle node) of simulation data generated by those models is different from the corresponding value of the original data. As for other methods, they just partially mask original samples, for many values have not been masked (coincident point). Although the GAN model, the Encoder-Decoder model and the differential privacy method generated simulation data differs from the original data, we wonder if the generated data can be used in practice. Figure 6 and Fig. 7 have shown the statistical results generated by the GAN model; and the statistical results generated by two other methods (Encoder-Decoder and differential privacy) are shown in Fig. 10 and Fig. 11.

The two statistical results (Figs. 10 and 11) show that they are totally different from the original data on statistical characteristics. This implicitly indicates that they cannot be used in practice for analysis. In other words, those results prove the effectiveness of our idea.

5 Conclusion

In this paper, we proposed a solution to the challenge of preserving data confidentiality while retaining the statistical characteristics of raw data. We utilized deep learning-based generative models (with various optimizers) to generate simulation data, and we conducted extensive experiments on real-world numeric Census and Environmental datasets. The results show the effectiveness of our idea.

Acknowledgments The authors would like to acknowledge the support provided by the National Key R&D Program of China (No.2018YFC1604000).

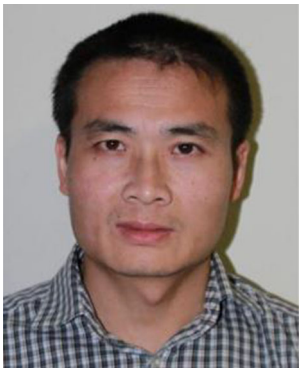
References

1. Rubin DB (1993) Discussion: Statistical disclosure limitation. J Off Stat 9
2. Min CL, Mitra R, Lazaridis E, An CL, Yong KG, Yap WS Data privacy preserving scheme using generalized linear models. Computers Security 2016
3. Gurjar SPS, Pasupuleti SK (2017) A privacy-preserving multi- keyword ranked search scheme over encrypted cloud data using mir-tree. In: Interna- tional conference on computing, analytics and security trends, pages 533–538
4. Andruszkiewicz P (2007) Optimization for mask scheme in privacy preserving data mining for association rules. In: International conference on rough sets and intelligent systems paradigms, pp 465–474
5. Willenborg L, De Waal T (2001) Elements of statistical disclosure control. Springer

6. Fienberg SE, Mcintyre J (2004) Data swapping: variations on a theme by dalenius and reiss. In International Workshop on Privacy in Statistical Databases, pages:14–29
7. Fuller WA (1993) Masking procedures for microdata disclosure limitation. J Off Stat 9(2)
8. Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014
9. Bengio Y (2009) Learning deep architectures for ai. Foundations Trends R in Machine Learning 2(1):1–127
10. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: International conference on neural information processing systems, pp 2672–2680
11. Martin Arjovsky, Soumith Chintala, and L'eon Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017
12. Barrow NJ, Campbell NA (1972) Methods of measuring residual value of fertilizers. Aus- tralian Journal of Experimental Agriculture 12(58):502–510
13. Simpson DG (1987) Minimum hellinger distance estimation for the analysis of count data. J Am Stat Assoc 82(399):802–807
14. Rubin DB (2009) Statistical Disclosure Limitation. Springer US
15. Li N, Li T, Venkatasubramanian S (2007) T-closeness: privacy beyond k-anonymity and l-diversity. In: Data engineering, 2007. ICDE 2007. IEEE 23rd international conference on. IEEE, pp 106–115
16. Dwork C (2008) Differential privacy: a survey of results. In: International conference on theory and applications of models of computation. Springer, pp 1–19
17. Van Tilborg HCA, Jajodia S (2014) Encyclopedia of cryptography and security. Springer Science & Business Media
18. Yang W, Li T, Jia H (2004) Simulation and experiment of machine vision guidance of agriculture vehicles. Transactions of the Chinese Society of Agricultural Engineering
19. Dormand JR, Prince PJ (1978) New runge-kutta algorithms for numerical simulation in dynamical astronomy. Celest Mech 18(3): 223–232
20. Stukowski A (2009) Visualization and analysis of atomistic simulation data with ovito-the open visualization tool. IEEE Trans Fuzzy Syst 23(6):2154–2162
21. Devia N, Weber R (2013) Generating crime data using agent-based simulation. Comput Environ Urban Syst 42(7):26–41
22. Phillips A, Cardelli L (2007) Efficient, correct simulation of biological processes in the stochastic pi-calculus. In International Conference on Computational Methods in Systems Biology, pages:184–199
23. Roe C, Meliopoulos AP, Meisel J, Overbye T (2008) Power system level impacts of plug-in hybrid electric vehicles using simulation data. In: Energy 2030 conference. Energy, pp 1–6, 2008
24. Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. Comput Therm Sci
25. Mirza M, Osindero S (2014) Conditional generative adversarial nets. Comput Therm Sci:2672–2680
26. Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. 2016
27. Dai F, Zhang D, Li J (2013) Encoder/decoder for privacy protection video with privacy region detection and scrambling. In: International conference on multimedia modeling. Springer, pp 525–527
28. Ismini Psychoula, Erinc Merdivan, Deepika Singh, Liming Chen, Feng Chen, Sten Hanke, Johannes Kropf, Andreas Holzinger, and Matthieu Geist. A deep learning ap- proach for privacy preservation in assisted living. arXiv preprint arXiv:1802.09359, 2018
29. Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In ICLR, 2016
30. Daskalakis C, Goldberg PW, Papadimitriou CH (2009) The complexity of computing a Nash equilibrium. ACM
31. Xu B, Wang N, Chen T, Li M (2015) Empirical evaluation of rectified activations in convolutional network. Comput Therm Sci
32. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1): 1929–1958
33. Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013), 2013
34. Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schul- man, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder arXiv preprint arXiv:1611.02731, 2016
35. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift, pp 448–456
36. Hardt M, Recht B, Singer Y (2015) Train faster, generalize better: stability of stochastic gradient descent. Mathematics
37. Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. 2017
38. Pizer SM, Amburn EP, Austin JD, Cromartie R, Geselowitz A, Greer T, ter Haar Romeny B, Zimmerman JB, Zuiderveld K (1987) Adap- tive histogram equalization and its variations. Computer vision, graphics, and image processing 39(3):355–368
39. Nowozin S, Cseke B, Tomioka R (2016) F-Gan: training generative neural samplers using variational divergence minimization. In: Advances in neural in- formation processing systems, pp 271–279
40. Bordes A, Bottou L, Gallinari P (2009) Sgdqn: Careful quasi-newton stochastic gradient descent. J Mach Learn Res 10(3): 1737–1754
41. T. Tieleman and G. Hinton. Rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA: neural networks for machine learning., 2012
42. Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res 12(7):257–269
43. Aleksandar Botev, Guy Lever, and David Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. 2016
44. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017) Improved training of wasserstein gans. In: Advances in neural information processing systems, pp 5767–5777

Wei Li is a Ph.D candidate of School Cyber Science and Engineering, Wuhan University. He had worked in the University of Massachusetts, Boston, and The Hong Kong Polytechnic University as visiting scholar. Email: auto_weili@whu.edu.cn

Pengqiu Meng is currently pursuing his master degree (2018-2020) in Computer Science at Washington University in St. Louis. His focal areas of study are data mining and artificial intelligence. He earned his bachelor degree (2014-2018) from Wuhan University, China and worked as a research assistant (2017-2017) at UC Berkeley, CAHL lab. Email: mengp@wustl.edu.



Hong Yi received his Ph.D. degree in Computer Science from University of California, Los Angeles, and he is now machine learning engineer in Sumo Logic Company. Email: yihongucla@gmail.com.

Xiaohui Cui received his Ph.D. degree in Computer Science from the University of Louisville in 2004, and he is a Professor of Computer Science in the Wuhan University. Email: xcui@whu.edu.cn.