

His-GAN: A histogram-based GAN model to improve data generation quality



Wei Li ^a, Wei Ding ^b, Rajani Sadasivam ^c, Xiaohui Cui ^{a,*}, Ping Chen ^{d,*}

^a School of Cyber Science and Engineering, Wuhan University, China

^b Mathematics and Science computer, University of Massachusetts Boston, USA

^c Quantitative Health Sciences and Medicine Division of Health Informatics and Implementation Science, University of Massachusetts Medical School, USA

^d Department of Engineering, University of Massachusetts Boston, USA

ARTICLE INFO

Article history:

Received 6 July 2018

Received in revised form 22 June 2019

Accepted 3 July 2019

Available online 24 July 2019

Keywords:

His-GAN

Histogram-based evaluation metric

Numeric simulation data generation

ABSTRACT

Generative Adversarial Network (GAN) has become an active research field due to its capability to generate quality simulation data. However, two consistent distributions (generated data distribution and original data distribution) produced by GAN cannot guarantee that generated data are always close to real data. Traditionally GAN is mainly applied to images, and it becomes more challenging for numeric datasets. In this paper, we propose a histogram-based GAN model (*His-GAN*). The purpose of our proposed model is to help GAN produce generated data with high quality. Specifically, we map generated data and original data into a histogram, then we count probability percentile on each bin and calculate dissimilarity with traditional f-divergence measures (e.g., Hellinger distance, Jensen–Shannon divergence) and Histogram Intersection Kernel. After that, we incorporate this dissimilarity score into training of the GAN model to update the generator's parameters to improve generated data quality. This is because the parameters have an influence on the generated data quality. Moreover, we revised GAN training process by feeding GAN model with one group of samples (these samples can come from one class or one cluster that hold similar characteristics) each time, so the final generated data could contain the characteristics from a single group to overcome the challenge of figuring out complex characteristics from mixed groups/clusters of data. In this way, we can generate data that is more indistinguishable from original data. We conduct extensive experiments to validate our idea with MNIST, CIFAR-10, and a real-world numeric dataset, and the results clearly show the effectiveness of our approach.

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

Generative Adversarial Network (GAN) (Mirza et al., 2014) has been demonstrated as the state-of-the-art generative model in various tasks of generating synthetic but realistic-like data (Chen et al., 2016; Yang et al., 2017; Zhu, Fidler, Urtasun, Lin, & Loy, 2017). A typical GAN usually consists of two main components, one is the discriminator, which could be regarded as a detective who could determine whether the data are original or generated; the other one is the generator, and it could be regarded as a forger who specializes in generating simulation data to fool the discriminator into accepting it (Creswell et al., 2018). The model iterates this process until the discriminator cannot distinguish whether the current data are generated by the generator or from

the original dataset. In this case, both distributions of generated data and original data would be deemed to approach consistent. The closer two distributions are, the better the effect is. The principal problem, after the completion of training, is that two consistent distributions (generated data distribution and original data distribution) do not mean the generated data being similar to original data.

The goal of GAN model is to transform a fixed, easy-to-sample distribution (e.g., Uniform distribution with $(-1, 1)$) into the real data distribution. In other words, the major concern of the generator is distribution transformation. However, the generated data quality details do not reflect in such a transformation process even though fooling discriminator successfully. For example, if one image contains two eyes, a nose and a lip, the generated image needs to contain the same components. As to generated data quality (e.g., the color of the eyes or the shape of the nose), it is usually not addressed by the training process (Mirza et al., 2014). That is to say, we do not know how well generated data quality is or how similar generated data and the original data are.

* Corresponding authors.

E-mail addresses: auto_weili@whu.edu.cn (W. Li), Wei.Ding@umb.edu (W. Ding), rajani.sadasivam@umassmed.edu (R. Sadasivam), xcui@whu.edu.cn (X. Cui), Ping.Chen@umb.edu (P. Chen).

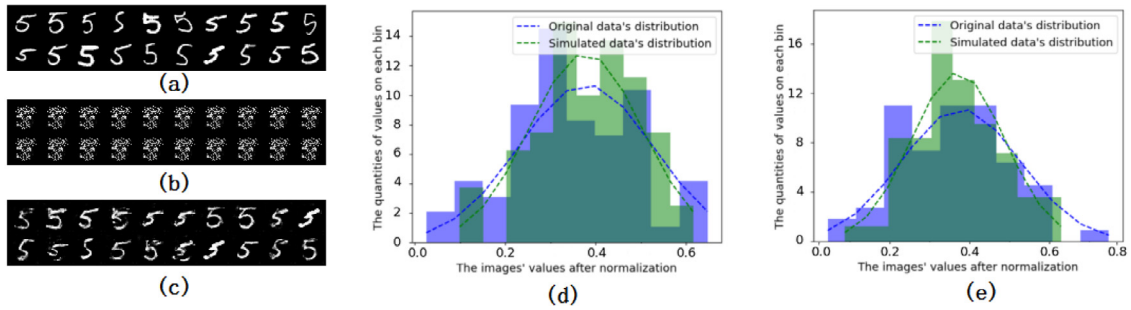


Fig. 1. Using a regular GAN model to produce generated data on MNIST dataset. The sub-figure (a) indicates the original data (Here we just use the handwritten figure '5'); while the sub-figure (b) and sub-figure (c) are generated images. The sub-figure (d) shows the relationship between the distribution of original data and that of generated data reflected by the sub-figure (b), while the sub-figure (e) shows generated data distribution for the sub-figure (c). We map all pixel values into the range of (0, 255), and we use an array of size 256 to save the pixel values. We count the statistical percentile of each entry in this array. After that, we use the *plt.hist()* and *MLA.normpdf()* functions with *bins = 10* to plot their distributions.

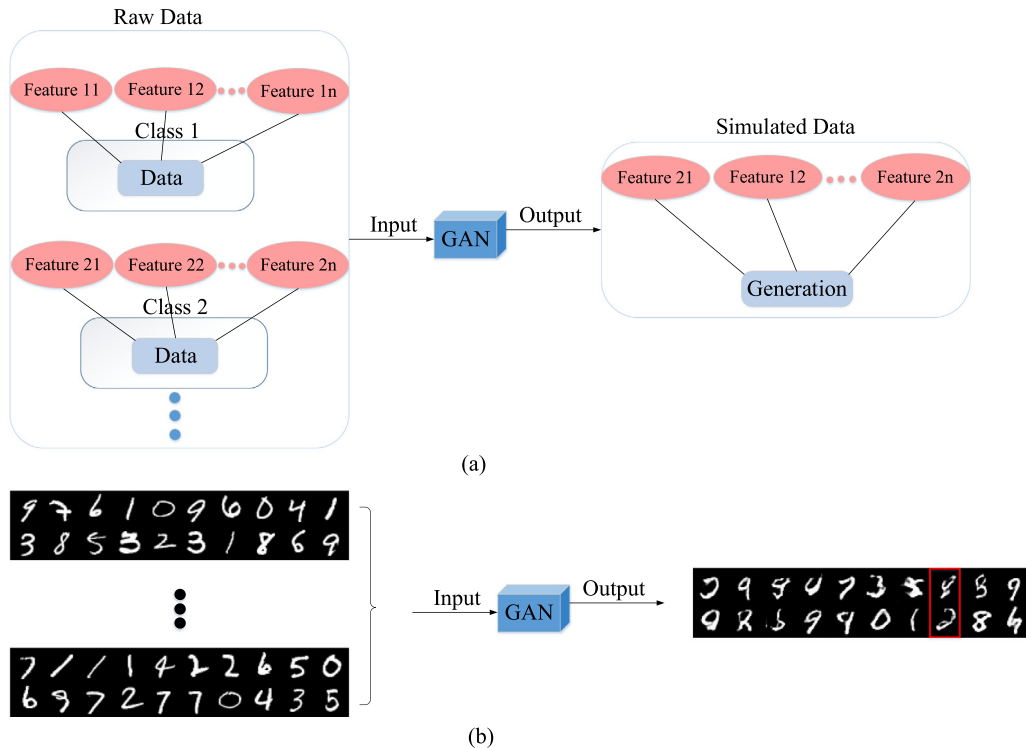


Fig. 2. The sub-figure (a) indicates an illustration of training a GAN model on multi-group data, and the sub-figure (b) indicates a specific example of training a GAN model on MNIST dataset. When we feed the multi-group data into the GAN model, it could output such generated data which may include different features or styles. In sub-figure (b), the red rectangle includes those generated images which are indistinguishable from the real ones. The top generated handwritten image holds the characteristics of figure '8' and figure '9'; as for the generated image at the below of the rectangle, it holds the characteristics of figure '2' and figure '0'. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To visually illustrate this problem, we want to show the relationship between the distribution and the quality of data in Fig. 1. In Fig. 1, the sub-figure (a) indicates the original data, the sub-figure (b) and sub-figure (c) indicate generated data produced by GAN. The sub-figure (d) indicates the distributions of the original data and generated data reflected by the sub-figure (b), and the sub-figure (e) indicates the same scenario with generated data distribution reflected by the sub-figure (c). Since the range of pixel values for an image is from 0 to 255, we use an array of size 256 to save the Hashed pixel values (e.g., $\text{array}[\text{pixel}] += 1$ if we map a pixel into the entry within the array). We repeat this process until all pixel values are saved. After that, we count the statistical percentile of each entry (e.g., $\frac{\text{entry}_i}{H \times W}$, entry_i indicates the value of *i*th entry and *H* indicates the height of the image and *W* indicates the weight of the image) to plot the corresponding histograms (See sub-figures (d) and (e)). From Fig. 1, we can see

that the distributions of original data and generated data are similar as shown in sub-figures (d) and (e). However, the generated images are blurry and indistinguishable (e.g., a generated handwritten image is similar to both figures '6' and '8', and more details are shown in Fig. 2) from real data, and the generation quality is not well. In other words, the two similar distributions cannot guarantee generated data being similar to the original data. Considering the significant influence and potential of the GAN model, it is critical to develop a new approach to address this challenge.

From Fig. 1, our insight is that the statistical gap shown in sub-figure (e) on each bin is smaller than that shown in sub-figure (d), which implicitly indicates that similarity between two datasets reflected by the statistical results in sub-figure(e) is better than that in sub-figure (d). This scenario inspires us integrating the statistical gap result into the training of GAN to improve the

generated data quality. Specifically, we measure the dissimilarity of generation and original data by mapping the two datasets into a histogram to calculate the relative frequentist statistics on each bin. After that, we combine the histogram-based measurement score with the original generator loss to update the generator's parameters via back-propagation, because the generator's parameters have an influence on the generated data quality. In this way, we name our new model *His-GAN* (discussed in Section 4).

Histogram retains more statistical information about a dataset, while the distribution (e.g., Gaussian distribution) is often characterized with just mean and variance in practice. Although we cannot always obtain real data distribution or generated data distribution (we have transformed a randomized space into the original data space), we can utilize the mean and variance which can be easily calculated to determine a distribution (Jaynes, 2003). Note that both the mean and the variance are the overall description of a dataset, it does not concern with detailed characteristics of a data sample. If generated distribution is similar to original data distribution, a GAN model would stop the training and output generated data. We can observe this in sub-figure (d) of Fig. 1. For histogram, each statistical percentile (bin) saves the statistical frequency of each entry of generated data. Different entries would hold different histograms (See sub-figure (e) of Fig. 1). If the histogram-based measurement score is integrated into training of GAN model to collectively determine stopping criteria, it will be required that the value of each entry of generated data is similar to that of original data. In this way, generated data quality can be improved. A detailed discussion about our approach is presented in Section 4.

In a histogram, each bin corresponds to a statistic of a sub-interval after we normalize and map all data into histogram. We calculate the statistical results (or probability percentile) on each bin. After that, we employ the traditional f-divergence metrics (Nowozin, Cseke, & Tomioka, 2016) (e.g., Hellinger distance Simpson, 1987, Jensen–Shannon (JS) divergence Manning & Schütze, 1999) and Histogram Intersection Kernel (HIK) method (Barla, Odone, & Verri, 2003; Pizer et al., 1987) to produce an objective value to measure dissimilarity between generated data and the original data in histogram. The smaller the value for f-divergence (the larger the value for HIK method), the higher generated data quality. More details are shown in Section 4.

Another improvement we made targets on different characteristics of generated data when we feed multi-group data into GAN model. A group can be a class or a cluster, whose data holds similar characteristics. On one hand, feeding multi-group samples into a GAN model improves the diversity of generated data; on the other hand, generating a complex distribution due to multiple groups is very challenging. To explore a new direction for this issue, we will investigate a group-based approach. This approach may hurt diversity of generated data, but provides a trade-off between a stable model and generated data quality. Multi-group samples may confuse the GAN model, because a category usually holds a specific distribution while the distribution of multi-group samples may be a combination of multiple distributions, and it is hard to be captured.

In summary, our study proposes a novel GAN variant, which helps GAN improve the generated data quality. Therefore, the major innovations and contributions of this paper are:

- This paper proposes the *His-GAN*, which trains the GAN by combining the histogram-based measurement score with the original generator loss to update the generator's parameters. In this way, the generated data quality is significantly improved.
- We utilize group-based approach to balance between generated data quality and a stable model.

- Through extensive experiments, we demonstrate the effectiveness of our approach.

The rest part of this paper is organized as follows. In Section 2 we discuss related work. We discuss the challenges of evaluating generated data quality in Section 3 and present our main ideas in Section 4. In Section 5 we will show our experiment results, and we conclude our work in Section 6.

2. Related work

The GAN model has shown impressive generative capabilities since its invention and has been widely used for various important machine learning tasks (Li et al., 2018). However, evaluating generation quality has been a serious challenge. In Salimans et al. (2016), Salimans et al. used a web interface to ask annotators to distinguish which sample is generated and which is real. However, this approach cannot be generalized since it was still limited to the image recognition and just used a program to evaluate whether a picture is acceptable or not. Gerhard et al. focused on qualitative comparison (such as comparing the visual quality). However, this technology was subjective and possibly misleading (Gerhard, Wichmann, & Bethge, 2013). Theis, Oord, and Bethge (2015) enumerated a variety of criteria (e.g., average log-likelihood, Parzen window, visual fidelity) to evaluate generative models. However, the results from those criteria showed that evaluating generative model is a problem-specific or domain-specific task. Some good results appearing in one criterion cannot extrapolate to another, while the computation of some criteria (e.g., likelihood) is intractable.


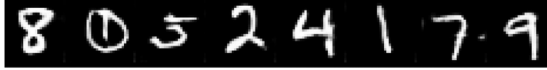


Maximum likelihood. The likelihood approach is widely considered as the default measure for quantifying generated data (Zhang et al., 2018). However, the calculation of likelihood is intractable, because the discrete data distribution has differential entropy of negative infinity, which causes the arbitrary high likelihood (Theis et al., 2015). In Wu, Burda, Salakhutdinov, and Grosse (2016), the study adds the optimal standard deviation of the noise to a generative model when maximizing likelihood is around 0.1 to each pixel in the generated image. This is a very high amount of noise, and the researchers do not add the noise on which they report likelihood numbers when they report the model's samples. In other words, the added noise can make the maximum likelihood approach work while it is incorrect for the problem (Arjovsky, Chintala, & Bottou, 2017). In addition, the generated results would be affected by those noises and cannot represent the true difference between the two distributions.

Maximum Mean Discrepancy (MMD) (Gretton, Borgwardt, Rasch, Schölkopf, & Smola, 2012). Lopezpez and Oquab (2016) used the Maximum Mean Discrepancy to evaluate the quality of generated image. The equation is defined as follows:

$$MMD[p, q] = (E_{p, q}[k(x, x') - 2k(x, y) + k(y, y')])^{\frac{1}{2}} \quad (1)$$

In Eq. (1), x, x' are random variables and independent to each other and they follow the distribution p . y, y' are also random variables and independent to each other and they follow the distribution q . p and q are two different distributions from two datasets. The goal of this equation is to find out whether the two distributions (p and q) are similar or not. The equation searches a special continuous function f that can maximize the mean discrepancy on two distributions to evaluate whether the two datasets are similar or not. The smaller the discrepancy is, the more similar the two distributions are. However, the MMD calculations are unstable within the GAN model. Let us take a look at an example in Table 1.

Table 1
Using MMD to assess the quality of generated data on MNIST dataset.

	Generated data	MMD
Id = 1		0.953
Id = 2		0.4595
Id = 3		0.0109
Id = 4		0.3192

From Table 1, we can see that we obtain high quality generation (Id = 2), but the MMD score is relatively high. The scenario (Id = 4) in which the quality of generated data is not as good as the Id = 2, with a low score. In other words, the MMD measure is unstable.

3. Preliminaries

3.1. Generative adversarial network

In this section, we formally present GAN model as below to establish the continuity. The GAN model was first proposed by Goodfellow (Mirza et al., 2014) as a novel generative model to simultaneously train a generator and a discriminator using the following novel loss function:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2)$$

In Eq. (2), x comes from a distribution $p_r(x)$ underlying the raw dataset and z comes from a pre-defined noise distribution $p_z(z)$ which is usually an easy-to-sample distribution, e.g., Uniform distribution or Gaussian distribution. The principal kernel of Eq. (2) is that the model can use the value of $D(G(z))$ that feeds the generator's outputs into the discriminator to generate training signals to update the generator's parameters. In this manner, the generator is able to fool the discriminator into accepting its outputs as real data. The generator G is modeled so that it can transform the noise z into the realistic-like data. In general, the generator G builds a mapping function from $p_z(z)$ to data space $p_r(x)$, and the discriminator would output a single score sitting in the range $[0, 1]$ to indicate whether the current data are from the generator or the raw dataset. Generally, a score closer to 0 indicates that the current data are generated by the generator with higher probability, while a score closer to 1 indicates that the current data come from the raw dataset with higher probability. We repeat this training process until both the discriminator and the generator reach a Nash equilibrium (Daskalakis, Goldberg, & Papadimitriou, 2009) when $p_G(z) = p_r(x) = 0.5$.

The characteristics of Eq. (2) are that $\max_D V(G, D)$ represents the difference between p_G and p_r when G is fixed, and $\min_G V(G, D)$ indicates that the generator G tries to minimize the difference between the two distributions when D is fixed. Hence, the optimization problem for the discriminator can be formulated as follows.

$$\max_D (\mathbb{E}_{x \sim p_r(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (3)$$

The discriminator D minimizes the score it assigns to generated data $G(z)$ by minimizing $D(G(z))$, and maximizes the score it assigns to the raw data x by maximizing $D(x)$. The generator G tries to fool the discriminator D into accepting its outputs as the

real data by maximizing its score $D(G(z))$, and this is achieved by optimizing the function $(\min_G (\mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]))$ in the generator. D and G are trained adversarially by competing with each other. As to other GAN variants (e.g., Least Square GAN (LSGAN) Mao et al., 2017, Wasserstein GAN (WGAN) Arjovsky et al., 2017, Improved WGAN Gulrajani, Ahmed, Arjovsky, Dumoulin, & Courville, 2017 and DeLiGAN Gurumurthy, Sarvadevabhatla, & Babu, 2017), they all adopt the same generating mechanism. From the data generation process, it is clear that the major challenge of generator is how to transform a distribution into another distribution to fool the discriminator into accepting simulation data as real data. However, close distribution does not always mean high generation quality (See Fig. 1).

3.2. Challenges of generating simulation data with GAN model

The training process of a GAN model is a black-box essentially. The generated data produced by GAN could consist of different characteristics or styles of different categorical data when we feed multi-group data into a GAN model. There is an example shown in Fig. 2.

In sub-figure (a), supposing those data belong to group 1 with n features, named $feature_{11}, feature_{12}, \dots, feature_{1n}$ respectively. Another set of data belonging to group 2 also have n features, named $feature_{21}, feature_{22}, \dots, feature_{2n}$. These features are different from each other. The generated data produced by a GAN model lies at the right side. The GAN model could mix all features together during training as we feed multi-group data into the model, so generated data may include different features. The sub-figure (b) shows a specific example. We input handwritten figures into a GAN model, and features are styles and shapes. In sub-figure (b), the red rectangle indicates those indistinguishable generated images from the real ones. The top generated handwritten image holds the characteristics of figure '8' and figure '9'; as for the generated image below the rectangle, it holds the characteristics of figure '2' and figure '0'. Especially, if this dataset belongs to the numeric dataset, the same problem still exists and it is more hard to distinguish generated data.

Conditional GAN (Mehdi Mirza, 2014) tries to distinguish generated data by producing class label with extra information that has been fed into both discriminator and generator. It integrates noise and extra information into a new vector, and puts it into the GAN model for training. However, often generated data is not consistent with the corresponding class label. One example is shown in Fig. 3.

In Fig. 3, each row represents a category from 0 to 9 (top to bottom). Obviously, generated data are not consistent with the corresponding class label. Also, the indistinguishable generated data are still inevitable.

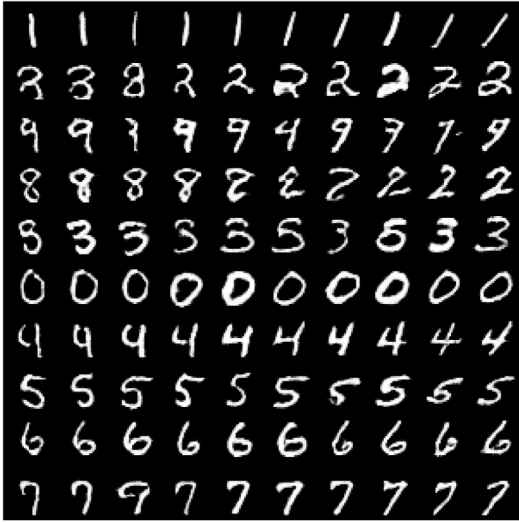


Fig. 3. Each row represents a category from 0 to 9 (top to bottom).

4. GAN-based modification and statistical similarity evaluation

In this section, we propose our statistical evaluation metric and the *His*-GAN model. As discussed in Section 1, the major innovation of our approach is to improve the generated data quality by combining the histogram-based measurement score with the original generator loss to update the generator's parameters. To this end, there are two important issues to be addressed. How to use histogram to measure the dissimilarity between generated data and original data, and how to integrate the measurement score into the GAN training process.

4.1. The histogram-based similarity measure for GAN model

In a GAN model, if $D(x) \approx D(G(z)) \approx 0.5$, the discriminator would deem that the distribution of original data is similar to that of generated data. However, similar distribution does not always mean high generation quality (See Fig. 1). This is because the distribution is often oversimplified by being represented with mean and variance when we map the data into the distribution (e.g., Gaussian distribution) (Jaynes, 2003). Note that both mean and variance are the overall description of a data sample, it does not concern with detailed characteristics of a data sample. Missing or replacing several data points often do not change the mean and the variance. On the other hand, histogram retains more information about a dataset. If taking face images as an example, a specific image can be saved in the form of a numerical array with an upper-bound and a lower-bound. All entries in this array determine what an image sample is because each entry represents a specific pixel (Zhang et al., 2016). Apparently current GAN models can-not reach low-level granularity details such as color or shape of generated eyes. It is noticed that such details (e.g., color of eyes) can be represented by statistical percentile or frequency information contained in a histogram. If two images are different, the frequencies or statistical percentiles would be different. The two statistical frequencies would be equivalent if the two datasets have same values at each sub-interval. Inspired by such a special characteristic of histogram, we can help GAN model produce high generated data quality by integrating the histogram-based measurement score into training of GAN. In the following discussion, we would introduce how to map the generated data and original data into the histogram and measure the dissimilarity, and demonstrate to integrate the measurement score into training of GAN.

4.2. How to map the data into histogram

The histogram function is shown as follows.

$$P_{hist}(i) = \frac{f_i}{N} \quad (4)$$

where N indicates size of a dataset and f_i indicates observed frequency of i th value in the dataset. $P_{hist}(i)$ indicates statistical result for this value. We map both generated data and original data with Eq. (4) and calculate corresponding statistical percentile to assess generated data quality.

For image datasets, we can use an array with size of 256 to save the pixel values, given that an image only contains 256 values (because the range of color in the computer is from 0 to 255). We directly use the array[pixel] + = 1 to count the corresponding pixel from the top-left entry to the bottom-right entry. After that, we plot the array in histogram. We can either define each pixel as the sub-interval or the range of pixel (e.g., Alec Radford & Chintala, 2015; Gulrajani et al., 2017) as the sub-interval.

For numeric datasets, we can use an array with arbitrary size to capture each number. For example, there is a dataset with two features ranging [[35, 60], [38, 65]]. Here we can use an array with size of 2 to save those values, array [0] indicates the range from 30 to 40 and its value is 2, while array [1] indicates the range from 60 to 70 and its value is also 2. Also, we can use an array with size of 10 to save those values, array[3] indicates the range from 30 to 40 and array[6] indicates the range from 60 to 70. From a statistical perspective, we do not care where the value is from, we just count this value in histogram.

In the traditional histogram similarity calculation, the results of a histogram would be the same when two images just contain two colors (e.g., half black and half white), even though the areas of colors in the two images are inverse to each other (e.g., the one is top-black and bottom-white and another one is top-white and bottom-black). In this case, histogram fails to assess similarity of the two images because it is 50% black and white. However, such a scenario is hard to occur in the GAN model. Those generated data contain noise at most and cannot be the inverse or other shape in practice. Equivalently, the generation for the numeric dataset (we take the same dataset ([[35, 60], [38, 65]]) as the example) is similar to the image dataset. The generated data could be [[30.5, 61], [36, 68]] (the similar generated data) or [[40, 45], [50, 55]] (the dissimilar generated data), and cannot be the inverse (e.g., [[61, 30.5], [68, 36]]), for GAN model specializes in generating plausible generation.

4.3. How to measure similarity in histogram

Assuming there are three numeric datasets, $X = [0.11, 0.12, 0.13, 0.24, 0.25, 0.26, 0.265, 0.27, 0.36, 0.35]$, $Y = [0.14, 0.22, 0.25, 0.26, 0.27, 0.28, 0.29, 0.31, 0.33, 0.35]$ and $Z = [0.14, 0.12, 0.25, 0.26, 0.27, 0.28, 0.29, 0.264, 0.33, 0.39]$ (Assuming X is the original data and the others are the simulation data). Since the difference between the three datasets are very small (They are in decimals and the range of all numbers lies in (0.1, 0.4)), it is hard to directly judge which one (Y or Z) is similar to the original dataset (X). Thus, we map the three datasets into a histogram, and each sub-interval is probability percentile. Assuming the number of sub-intervals is 3 (all samples are counted in the 3 sub-intervals and the range of each sub-interval is 0.1), so $Q(X) = [0.3, 0.5, 0.2]$ ($3 \setminus 10, 5 \setminus 10, 2 \setminus 10$) and $Q(Y) = [0.1, 0.6, 0.3]$ and $Q(Z) = [0.2, 0.6, 0.2]$ respectively. In other words, we transform calculating similarity between two datasets (X and Y or Z) into calculating similarity between $Q(X)$ and $Q(Y)$ or $Q(X)$ and $Q(Z)$. The corresponding histogram is shown in Fig. 4.

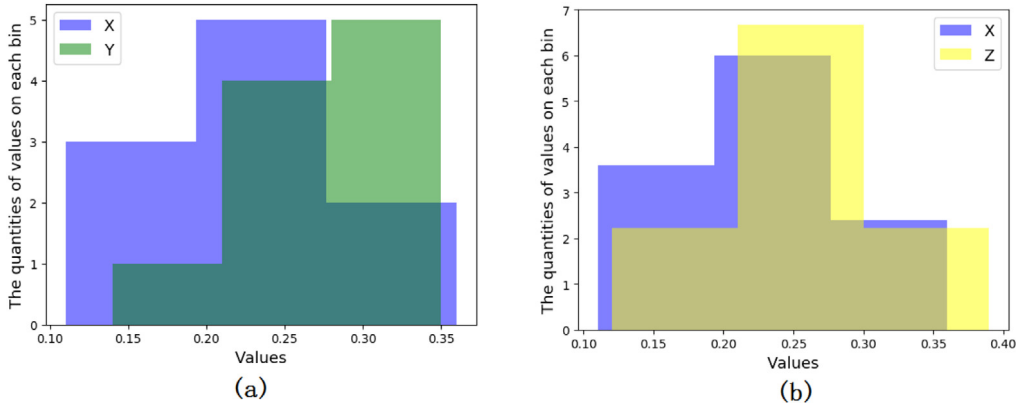


Fig. 4. An example to demonstrate histogram-based similarity measure. Sub-figure (a) reflects the statistical similarity between both X and Y , while sub-figure (b) reflects the statistical similarity between X and Z .

Table 2

Using f -divergence community (Hellinger Distance (HD), KL divergence, JS divergence, Wasserstein distance (WD)) and Histogram Intersection Kernel (HIK) method to calculate similarity between X and Y or Z . The first four methods are from SCIPY, and we utilize the OpenCV to calculate HIK results.

ID	HD	KL	JS	WD	HIK
$P_{(Q(X),Q(Y))}$	0.7959	0.2579	0.0622	0.6667	7.0
$P_{(Q(X),Q(Z))}$	0.2707	0.0305	0.0073	0.6667	9.0

In Fig. 4, we find that the statistical gap between X and Z is smaller than X and Y (In other words, dataset Z is closer to dataset X than Y). To measure such statistical gap, we need to develop an objective single score to reflect which dataset (Y or Z) is more similar to X . Here we use the f -divergence community, such as Kullback–Leibler (KL) divergence ($D_{KL}(P \parallel Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}$) (Joyce, 2011), Hellinger distance ($H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^k (\sqrt{p_i} - \sqrt{q_i})^2}$) (Simpson, 1987), Jensen–Shannon (JS) divergence ($JS(P \parallel Q) = 0.5 \times D_{KL}(P \parallel \frac{P+Q}{2}) + 0.5 \times D_{KL}(Q \parallel \frac{P+Q}{2})$) (Manning & Schütze, 1999) and Wasserstein distance ($WD(P, Q) = \frac{\sum_i \sum_j d_{ij} f_{ij}}{\sum_i \sum_j f_{ij}}$) (Dobrushin, 1970), and Histogram Intersection Kernel (HIK) (Barla et al., 2003) ($HIK_{\cap}(a, b) = \sum_{i=1}^n \min(a_i, b_i)$ where a and b are two histograms with n bins each, and the mechanism of HIK is the intersection of minimum statistical quantities (or histogram intersection) on each bin of two histograms.) to measure similarity. The results are shown in Table 2.

In Table 2, the HD, KL, JS and Histogram Intersection Kernel reflect the scenario where the dataset Z is more similar to dataset X than Y , while the WD fails to measure such similarity fluctuation because the two results are same.

Note that it is unfair in Table 2 if we directly compare the two results (e.g., comparing 0.7959 with 0.2579), because the value interval is different for different distance metrics. For example, the maximum value for HD metric is 0.7959 and 0.2579 for KL metric when they take the same dataset (e.g., dataset Y). Thus, we use the normalization method ($Norm_i = \frac{X_i - \min_value}{\max_value - \min_value}$, X_i indicates a specific entry) (Wolfensberger, Nirje, Olshansky, Perske, & Roos, 1972) to normalize these distances and compare those metrics. The normalization method is a non-dimensional method that can transform an absolute value into a relative value. In addition, we assume that the generated data with high quality follow the same distribution as the original data, so the minimum value for f -divergence community (HD, KL, JS and WD) using the normalization method is set to 0, and the maximum value is assumed as the generated noise (e.g., 0.7959 for the HD

metric). However, Histogram Intersection Kernel is opposite to the f -divergence (the larger the result, the better similarity), so the maximum value for Histogram Intersection Kernel using the normalization method is set to 10, and the minimum value is also assumed as the generated noise (e.g., 7.0 in this case).

The five metrics' results after normalizing are 0.3401 (e.g., $\frac{0.2707-0}{0.7959-0}$, HD), 0.1183 (KL), 0.1174 (JS), 1 (WD) and 0.6667 (e.g., $\frac{9.0-7.0}{10.0-7.0}$, Histogram Intersection Kernel), respectively. The first four metrics have the same evaluation standard (the smaller the result, the better similarity), we conclude that the JS (0.1174) has a better evaluation than HD (0.3401), KL (0.1183) and WD (1). In next subsection, we introduce how to integrate the measurement score into training of GAN.

4.4. His-GAN

Assuming we have obtained measurement score which can reflect the similarity between generated data and original data, we then incorporate this score into training process of the GAN model. Let us refer back to the Eq. (2), the generator fools the discriminator by maximizing its score $D(G(z))$, and this is achieved by optimizing the function ($\min_G (E_{z \sim p_z(z)} [\log(1 - D(G(z)))])$) in the process of training the generator. As a deep learning model, the values from this function are used to update the generator's parameters, and these parameters have an influence on generated data quality. In this way, we combine the measurement score with the generator loss, and use this combination to update the generator's parameters. The measurement function is shown in Eq. (5), and the new function of generator within His-GAN is shown in Eq. (6).

$$hist(x, G(z)) = P(x) - P(G(z)), \text{ s.t.}, P(x) = \frac{f_i}{N} \quad (5)$$

$$\min_G V(G^*, D) = \min_G (E_{z \sim p_z(z)} [\log(1 - D(G(z)))] + hist(x, G(z))) \quad (6)$$

where $P(\cdot)$ indicates the histogram function in which f_i indicates observed frequency of i th value in a batch dataset and N indicates the size of this dataset. z is sampled from an easy-to-sample distribution and x is from original dataset. From Eq. (5), we can see that the term $hist$ is a function of x if we transform $G(z)$ into x during training. Note that many works adopt the similar idea to modify the GAN model (e.g., using $L1$ - loss or MSE) to achieve their goals according to different tasks. We take the $pix2pix$ ($L1$ - loss) (Isola, Zhu, Zhou, & Efros, 2017) and context encoder (MSE) (Pathak, Krähenbühl, Donahue, Darrell, & Efros,

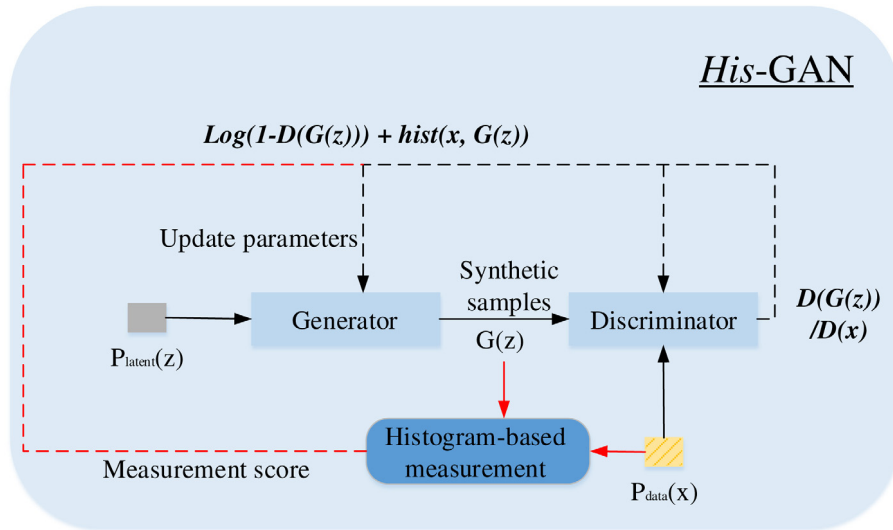


Fig. 5. The *His-GAN* model architecture. We calculate histogram-based measurement score after generating the simulation data. Then, we integrate this score into training of GAN. The red dash-line indicates the integration process. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2016) as the example, and give the **difference** between the two studies and our approach as follows.

For *pix2pix* study, the extra loss $\mathcal{L} = \lambda \times \mathbb{E}_{x,y,z} \|y - G(x, z)\|$ where x indicates the edge map and y indicates the original map as well as z indicates a random noise, given that *pix2pix* study focuses on image translation (e.g., an edge map is rendered as an RGB image). In this study, \mathcal{L} is an auxiliary factor and it works with “U-Net” to translate images. For *context encoder* study, $\mathcal{L} = \|\hat{M} \odot (x - F((1 - \hat{M} \odot x))\|_2$ where x indicates ground truth image and F indicates the context encoder as well as \odot is the element-wise product operation, because *context encoder* focuses on image inpainting. The extra loss \mathcal{L} is still an auxiliary factor and it works with encoder–decoder pipeline to inpaint images. For our study, the extra loss indicates *hist*. It is a vital factor and is combined with the generator loss to update a generator’s parameters.

In the early training stage, the generator basically outputs the “noise” instances, so the discriminator has been easily driven into the optimal status under such a scenario in which the discriminator can always distinguish that the current data are from the real dataset rather than the generator ($D(x)$ outputs 1 while the value of $D(G(z))$ approaches to 0). In this way, $\text{Loss}_G = \log[1 - D(G(z))] \approx 0$, it means that the vanishing gradients arise. If the generator cannot get the meaningful gradients, the parameters of the generator would not be updated so generated data quality cannot be guaranteed, given that the network’s parameters have an influence on generated data quality. Our function can avoid this case. We state the benefit of Eq. (6) properly in the following explanation.

- When the discriminator has been driven into the optimal status, $\log(1 - D(G(z)))$ still outputs 0. However, *hist* outputs a relative large value (we adopt the JS-divergence to calculate histogram measurement score), given that the quality of generated data is far from “real”. In this way, $V(G^*, D) > 0$, and the generator can update its parameters with *hist*.
- When GAN model has been trained successfully, i.e., $D(x) \approx D(G(z)) \approx 0.5$, the term $\log(1 - D(G(z)))$ implicitly approaches to $\log(0.5)$. The value of *hist* is very small (we can view it as 0) under such a scenario. This is because GAN model has generated the realistic-like data, and histogram-based measurement score approaches to 0. The smaller the JS-divergence value is, the better similarity

between generated data and the original data is. Eq. (6) has been reformulated into $\min_G (\mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))])$ as the regular GAN model if GAN model reaches to the Nash Equilibrium. It implicitly indicates the function Eq. (6) is convergent.

In addition, we present the pseudo-codes of the algorithm in Algorithm 1 and illustrate the process of integrating the measurement score into training of GAN in Fig. 5. The goal of Algorithm 1 is still pursuing to minimize the generator G while maximizing the discriminator D , so the discriminator D is unchanged. The difference from training of regular GAN model is the generator G . We calculate the histogram-based measurement score with f-divergence (e.g., JS-divergence) after generating the simulation data. After that, we incorporate the measurement score into the original generator loss to update the parameters of generator with back-propagation strategy. The parameters would have an influence on generated data quality. We can observe the performance of the *His-GAN* in the Section experiments.

4.5. Applying group-based strategy to GAN training process

When we feed multi-group data samples into GAN model, generated data holds characteristics from different groups. Although there are many GAN variants (e.g., Least Square GAN (LSGAN) Mao et al., 2017, Wasserstein GAN (WGAN) Arjovsky et al., 2017, Improved WGAN Gulrajani et al., 2017 and DeLiGAN Gurumurthy et al., 2017), their generating mechanism is similar to original GAN model, and faces the same challenge of mixed characteristics from different clusters or classes (See Fig. 6).

In Fig. 6, these four sub-figures show the images generated by WGAN (Arjovsky et al., 2017), Improved WGAN (Gulrajani et al., 2017), LSGAN (Mao et al., 2017), DeLiGAN (Gurumurthy et al., 2017), respectively. In each sub-figure, we use red rectangles to indicate the generated images which are indistinguishable from real ones (e.g., the generated image marked by the red rectangle shown in sub-figure (d) is similar to figures ‘7’ and ‘9’). It is clear that the model mixes characteristics of multi-group samples.

In general, data of the same cluster or class is closely related to each other, and they have the same or similar features (e.g., shape

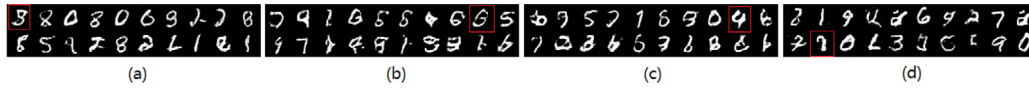


Fig. 6. These four sub-figures illustrate data generated by WGAN (Arjovsky et al., 2017), Improved WGAN (Gulrajani et al., 2017), LSGAN (Mao et al., 2017), DeLiGAN (Gurumurthy et al., 2017), respectively. The red rectangle indicates the indistinguishable image from the real one.

Table 3

Using Hellinger distance ($HD(P_{data}(x), p_z(z))$), KL divergence ($KL(P_{data}(x) \parallel p_z(z))$), JS divergence ($0.5 * P_{data}(x) \parallel p_z(z) + 0.5 * (P_z(z) \parallel P_{data}(x))$), Wasserstein distance ($WD(P_{data}(x), p_z(z))$) and Histogram Intersection Kernel ($HIK_{\gamma}(P_{data}(x), p_z(z))$) to assess the quality of generated data on the MNIST. The first row indicates a scenario where the training epoch = 1, and the second row indicates the training epoch = 10, and the third row indicates the training epoch = 20, and the fourth row indicates the training epoch = 30. The last row indicates the original images. The right results reflect similarity between the original data and generated data, after we map them into histogram. With the generation becoming clear, the scores gradually become smaller (HD, JS and WD) or larger (HIK). Notice that those results belong to absolute distances so we need to normalize them for comparing those metrics, and the normalization results are shown in Table 4.

Generated image/ Original image	HD	KL	JS	WD	HIK
	0.6598	inf	0.3766	0.0055	4 214
	0.5432	inf	0.2482	0.0041	7 726
	0.3456	inf	0.102	0.0019	13 096
	0.2763	0.2866	0.0663	0.0011	14 778

Algorithm 1 His-GAN.

Input:

Raw dataset;
noise $z, p_z(z)$;

Output:

Generated data.

Adam optimizer and BCE loss function

for number of iterations do

- Sampling minibatch of m noise samples z^1, \dots, z^m from Uniform distribution with $(-1,1)$.
- Sampling minibatch of m original samples x^1, \dots, x^m from the real dataset.
- Updating the discriminator’s parameters by ascending its stochastic gradient.
- $\nabla_{\theta_D} \frac{1}{m} \sum_1^m \{\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))\}$.
- Sampling minibatch of m noise samples z^1, \dots, z^m .
- Calculating histogram-based measurement score using Eq. (5).
- Updating the generator’s parameters by descending its stochastic gradient using Eq. (6).
- $\nabla_{\theta_G} \frac{1}{m} \sum_1^m \{\log(1 - D(G(z^{(i)})))\} + hist(x, G(z))$.

end for

GAN model after training this GAN model with a sub-group of data samples, because hyperparameters should be group-specific. We repeat the training process until all clusters are used. This process of feeding multi-group data samples (e.g., one group at one time) can avoid generating indistinguishable samples. See Fig. 6.

5. Experiments

To validate our approach, three datasets, MNIST, CIFAR-10, and a real-world numeric dataset (Smoking Cessation dataset), are studied. MNIST and CIFAR-10 are image datasets, and they are widely used in machine learning and computer vision. The Smoking Cessation dataset is a questionnaire dataset collected in medical domain.

5.1. MNIST dataset

We first produce simulation images of MNIST dataset where each sample is a gray image with 28×28 size. This dataset has 10 categories, which are ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, respectively. Before applying our proposed His-GAN to MNIST, we should make sure that the histogram-based measurement score can objectively reflect the dissimilarity between generated images and original images. Although we describe the advantage of histogram-based measurement in previous section, it is necessary to further demonstrate the effectiveness of histogram on assessing the generated data quality in experiment. Here, we keep the generated images produced by DCGAN after each completion of training epoch, and use the evaluation metric (HD, KL, JS, WD and HIK) to calculate the similarity between the generated images and original images after mapping these images into the histogram. The generated images and the measurement scores calculated by histogram-based measurement metric are shown in Table 3. In Table 3, the first four rows of the left part are generated images, and the right part of the first four rows shows the assessing results.

or style or color). Features are different when data is from different clusters or classes. Data within a cluster is of high cohesion but low coupling in different clusters.

Our idea is to feed data from one group (e.g., a cluster of samples that are close and similar to each other) each time to GAN model for easier training. We first group the original data into several sub-groups. Then we input the sub-groups into GAN model one by one. Note that we re-initialize hyperparameters of

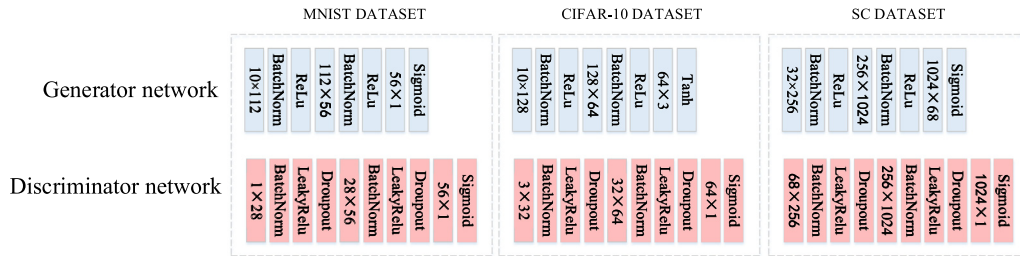


Fig. 7. Architectural details of GAN model. Learning rate is 0.0002, and weights of our model are set to Normal(0.0, 0.02) and biases are set to (0.0). We use Adam gradient method (Kingma & Ba, 2014) and Binary Cross Entropy (Kroese, Rubinstein, Cohen, Porotsky, & Taimre, 2013) to update the parameters of generator and discriminator. We set LeakyRelu (Xu, Wang, Chen, & Li, 2015) as 0.02 and Dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) as 0.5. Activation function of last layer for generator is Sigmoid (Han & Moraga, 1995) for MNIST and Smoking Cessation datasets, and that is Tanh for CIFAR-10 dataset. In MS dataset, the input channel at the first layer for discriminator is set to 68, for this dataset consists of 68 features. In all datasets, the batch size has been set to 20.

Table 4

We use the normalization method to transform those absolute distances shown in Table 3 into the relative distances. Notice that the maximum value for HIK is 18724 (width * height) and the minimum value is 4214. As for other four metrics, the maximum value lies in the first row (e.g., 0.6598 for HD) and the minimum value is 0. We then test those metrics using only one generated image (the red rectangle marked in Table 3), and the results are shown in Table 5.

HD	KL divergence	JS divergence	WD	HIK
0.8233	inf	0.6591	0.7455	0.242
0.5238	inf	0.2708	0.3455	0.6121
0.4188	inf	0.176	0.2	0.728

Table 5

We pick up only one generated image (red rectangle marked in Table 3). After we map them into histogram, we use the same metrics to calculate similarity between the generated image and the original image. The results show the same trend for similarity like Table 3. Since those results still belong to the absolute distances so we need to normalize them for comparing those metrics, and the normalization results are shown in Table 6.

HD	KL divergence	JS divergence	WD	HIK
0.9513	inf	0.6409	0.0064	37
0.7392	inf	0.4267	0.0048	160
0.5285	inf	0.2099	0.0027	429
0.4426	inf	0.1465	0.0018	445

Table 6

We use the normalization method to transform those absolute distances shown in Table 5 into the relative distances. Here the maximum value for HIK is 784.

HD	KL divergence	JS divergence	WD	HIK
0.777	inf	0.6658	0.75	0.1647
0.5556	inf	0.3275	0.4219	0.5248
0.4653	inf	0.2286	0.2813	0.5462

Although the first row shows the noise in Table 3, it still belongs to generated data produced by the generator and it is necessary to assess its similarity with original data by using a certain value, for we want to know how similar or dissimilar the two datasets are. The KL divergence cannot reflect such fluctuation as it outputs the *inf* (which means infinite). The Hellinger distance, JS divergence, Wasserstein distance and Histogram Intersection Kernel can royally reflect the fluctuation. However, those four metrics are essentially different.

Since some values in probability percentile are 0 (these images could lack some pixels), it causes the $\log(0)$. Thus, the results are *inf* for KL divergence. As to JS divergence, its denominator is $\frac{P_{data}(x)+P_z(z)}{2}$, and it can avoid such scenario in which probability is 0. Thus, JS divergence avoids the $\log(0)$.

We continue to test those metrics with only one generated image (It has been marked by red rectangle in Table 3), and the results are shown in Table 5. The corresponding statistical results are shown in Fig. 8 (we adopt the function of *plt.hist()* with *bins* =

Table 7

We map generated data shown in Fig. 9 into histogram, and measure their similarity to original data. The score clearly shows the improvement using our approach.

	HD	KL divergence	JS divergence	HIK
DCGAN	0.0832	0.0169	0.0047	18 338
His-GAN	0.031	0.0064	0.0015	18 465

10 and $\alpha = 0.2$). Fig. 8 shows that probability similarity on each sub-interval is gradually approaching (from sub-figure (a) to sub-figure (c)), with the generated images gradually becoming less blurry (See Table 3).

Fig. 8 shows the statistical results of both generated data and the original data. With these generated images gradually becoming less blurry row by row in Table 3, the values of Hellinger distance and JS divergence and Wasserstein distance are becoming smaller (histogram intersection kernel is becoming larger) (See Tables 3 and 5).

In this case, the histogram-based measurement shows that it can objectively reflect the similarity between generated data and original data. We then incorporate the measurement score into the GAN training process shown in Algorithm 1. The detailed architecture is shown in the left part of Fig. 7, and the histogram-based measurement score is calculated by JS-divergence. In the experiments, we adopt regular DCGAN as the baseline, because generated data produced by most GAN variants hold the similar quality (See Fig. 6). The size of training epoch has been set to the same number (e.g., *Epoch* = 30), and generated data are shown in Fig. 9.

In Fig. 9, sub-figure (b) shows images generated by His-GAN, while images shown in sub-figure (c) are from DCGAN. Sub-figure (a) reflects losses from DCGAN's generator and His-GAN's generator. Early in the training of the His-GAN model, generated data are very different from original data, so the His-GAN's generator loss is very large. As training continues, generated data gradually resemble original data (score ≈ 0 , Table 3 shows such a trend from row 1th to row 4th), and the measurement score also becomes smaller. His-GAN's generator loss would approach to DCGAN's generator loss under such a scenario. The two models would converge to such a point where $D(x) \approx D(G(z)) \approx 0.5$. By comparing, we found that the quantity of indistinguishable generated data from His-GAN is less than DCGAN. In other words, generated data quality of His-GAN is better than DCGAN. For validating effectiveness of our approach, we measure similarity with f-divergence and Histogram Intersection Kernel in histogram (*bins* = 10), and histogram-based measurement scores are shown in Table 7.

From Fig. 9 and Table 7, we can see that the generated data quality is significantly improved. However, the generated images

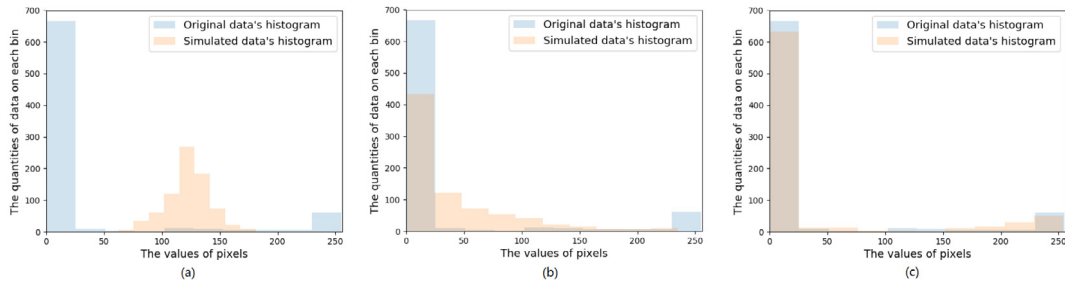


Fig. 8. The sub-figure (a) corresponds to generated data at the first row in Table 3. The sub-figure (b) and sub-figure (c) correspond to generated data at the second row and the fourth row in the same table. Here we use the *plt.hist()* function to obtain histogram. With the appearance of generated data becoming clear, the statistical results on each sub-interval are approaching.

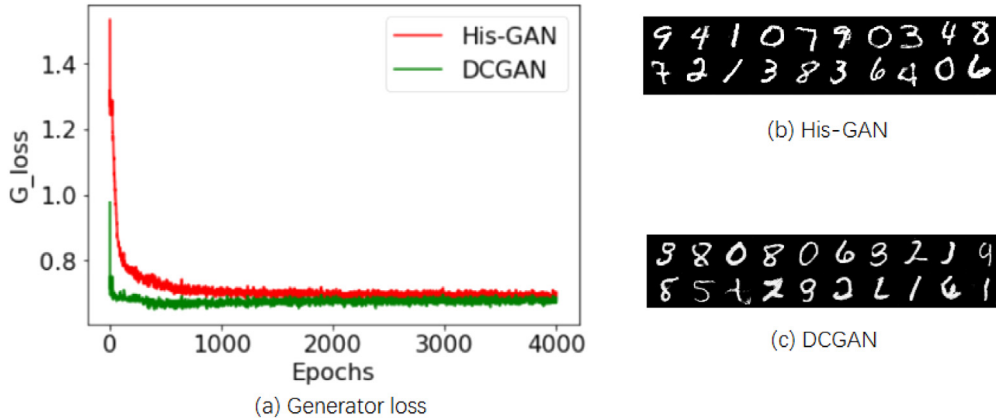


Fig. 9. Sub-figure (a) indicates generator loss for both *His-GAN* and *DCGAN*. Sub-figure (b) shows simulation data generated by *His-GAN*, while sub-figure (c) shows simulation data generated by *DCGAN*.

still exist the indistinguishable samples (e.g., row 2, column 1 and 8 in sub-figure (b)). We then apply the group-based strategy to search for a trade-off between the diversity and the generated data quality. The generated data are shown in Fig. 10. Apparently, those generations produced by our idea are better than other GAN-based models (See Figs. 3 and 6). The generations are incorrectly classified in CGAN or simulation images are not readable (e.g., the generated image with red rectangle in Fig. 6).

5.2. CIFAR-10 dataset

CIFAR-10 dataset is a color image dataset with 3*32*32 size, it also consists of 10 classes, which are ‘airplane’, ‘automobile’, ‘bird’, ‘cat’, ‘deer’, ‘dog’, ‘frog’, ‘horse’, ‘ship’, ‘truck’, respectively. Similar to MNIST dataset, we still need to validate the histogram-based measurement metric on assessing the similarity between generated data and original data before applying *His-GAN* to this case. We adopt same generating method to generate color simulation images, and the hyperparameters are shown in the middle part of Fig. 7. We directly map the generated and training samples into a histogram, and use the same metrics to calculate similarity. The results are shown in Table 8.

In Table 8, the first row indicates a scenario where training epoch = 1, and the second row indicates training epoch = 10, and the third row indicates training epoch = 20, and the fourth row indicates training epoch = 30. The last row indicates the original images (‘truck’). The right part reflects measuring results between original data and generated data. Like Table 3, KL fails to assess the quality of generated data while HD, JS, WD and HIK reflect such similarity fluctuation. We then use normalization method to calculate relative distances, and results are shown in Table 9.

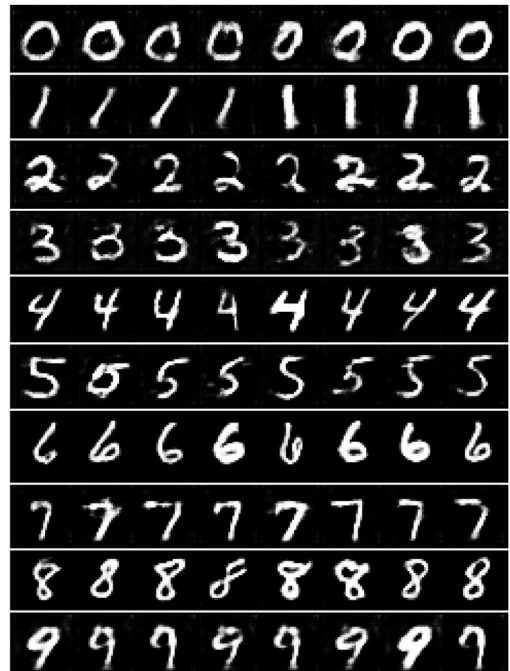


Fig. 10. We adopt the group-based strategy to train GAN model on MNIST dataset. We adopt a set of samples with same category (e.g., all samples belong to category 0) as the training data, and each row represents a set of generated data in which their categories are from 0 to 9 (top to bottom).

Notice that the maximum value in Table 9 for Histogram Intersection Kernel is 23940 (width * height) and the minimum

Table 8

Using GAN model to generate simulation data and using Hellinger distance ($HD(P_{data}(x), p_z(z))$), KL divergence ($KL(P_{data}(x) \parallel p_z(z))$), JS divergence ($0.5 * P_{data}(x) \parallel p_z(z) + 0.5 * (P_z(z) \parallel P_{data}(x))$), Wasserstein distance ($WD(P_{data}(x), p_z(z))$) and Histogram Intersection Kernel ($HIK_{\cap}(P_{data}(x), p_z(z))$) to assess the quality of generated data on the CIFAR-10 dataset.

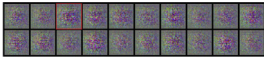

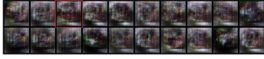
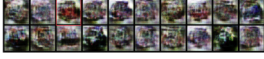

Generated image/ Original image	HD	KL	JS	WD	HIK
	0.5116	inf	0.2182	0.0034	11 114
	0.3216	inf	0.0867	0.0018	17 279
	0.2988	inf	0.0734	0.0014	18 094
	0.1828	inf	0.0288	0.0006	20 640
	Original data				

Table 9

We use the normalization method to transform those absolute distances shown in Table 8 into the relative distances.

HD	KL divergence	JS divergence	WD	HIK
0.6286	inf	0.3973	0.5294	0.4807
0.5841	inf	0.3364	0.4118	0.5442
0.3573	inf	0.132	0.1765	0.7427

Table 10

We pick up only one generated image (red rectangle marked in Table 8). After we map them into histogram, we use the same metrics to calculate similarity between the generated image and the original image. Results show the same trend for similarity like Table 8. Same as Table 9, we use normalization method to calculate relative distances for comparing, and results are shown in Table 11.

HD	KL divergence	JS divergence	WD	HIK
0.7156	inf	0.3551	0.0023	261
0.4064	inf	0.2446	0.0008	456
0.4041	inf	0.1731	0.0008	517
0.3986	inf	0.1395	0.0007	607

Table 11

We normalize absolute distances shown in Table 10 into relative distances. Here the maximum value for HIK is 1024.

HD	KL divergence	JS divergence	WD	HIK
0.5679	inf	0.6888	0.3478	0.2556
0.5647	inf	0.4875	0.3478	0.3355
0.557	inf	0.3928	0.3043	0.4535

value is 11114. As for other four metrics, the maximum value lies in the first row (e.g., 0.5116 for HD) and the minimum value is 0. We then test those metrics using only one generated image (the red rectangle marked in Table 8), and results are shown in Table 10. In addition, relative distances are shown in Table 11.

From the two sets of results, we can see that KL divergence fails to calculate similarity between generated data and original data, and it outputs *inf* in histogram, while Hellinger distance, JS divergence, Wasserstein distance and Histogram Intersection Kernel methods can royally reflect such similarity. In f-divergence community, we observe that JS divergence performs better than other three metrics (Hellinger, JS and Wasserstein) when we transform absolute distances (Tables 3, 5, 8 and 10) into relative distances (Tables 4, 6, 9 and 11), for JS divergence holds the minimum score. For Histogram Intersection Kernel, it also displays same similarity fluctuation (the score is becoming larger, with generated images becoming clearer). Thus, in image dataset, JS divergence and Histogram Intersection Kernel might be a good choice when normalizing absolute distances into relative distances, after mapping generated data and original data into histogram.

Table 12

We map generated data shown in Fig. 11 into histogram, and measure their similarity with original data. The score shows that our approach is effective.

	HD	KL divergence	JS divergence	HIK
DCGAN	0.1148	0.0516	0.01299	21 046
His-GAN	0.065	0.0167	0.0042	22 146

In the color image case, the histogram-based measurement also shows the effectiveness of assessing the similarity between generated data and original data. We then integrate histogram-based measurement score into training process of GAN model, which is shown in Algorithm 1 (*Epoch* = 30), to improve the generated data quality. We adopt the same architecture as Fig. 7, and the score is still from JS-divergence, given that JS-divergence outperforms other metrics (See Tables 9 and 11). We continue to compare *His*-GAN with regular DCGAN, and generated data are shown in Fig. 11. In Fig. 11, sub-figure (a) shows loss values of two generators from *His*-GAN and DCGAN, and both sub-figure (b) and sub-figure (c) display generated data produced by *His*-GAN and DCGAN, respectively. Trend of generator loss in CIFAR10 dataset is similar to MNIST. Early in training, *His*-GAN's generator loss is larger than DCGAN's generator loss. The difference between two loss curves is small in this case, because measurement score is very small (See first row of Table 8) but generator loss is relatively large. When training goes on, generated data gradually resemble original data, and measurement score becomes smaller (Table 8 shows such a scenario from row 1th to row 4th). From both sub-figure (b) and sub-figure (c), we can see that the quality of generated data produced by *His*-GAN is better than DCGAN. We also measure histogram similarity (*bins* = 10) with f-divergence and Histogram Intersection Kernel, and results are shown in Table 12. Next, we would apply our approach to the numeric dataset.

5.3. Smoking cessation dataset

The Smoking Cessation dataset consists of smokers who were invited into the Share2Quit program (Rebecca Kinney, Sowmya Rao, Sadasivam, Erik Volz, & Thomas Houston, 2013). In this program, smokers were encouraged and incentivized to participate in an online peer recruitment study to try to invite more smokers to participate in an online tobacco cessation intervention (Sadasivam et al., 2016). The goal of this project is to invite more smokers to participate in our program while spending less cost. So, the more smokers are recruited with less cost, the more excellent the recruiter. In this program, each user was given 30 days to recruit family and friends smokers. A successful

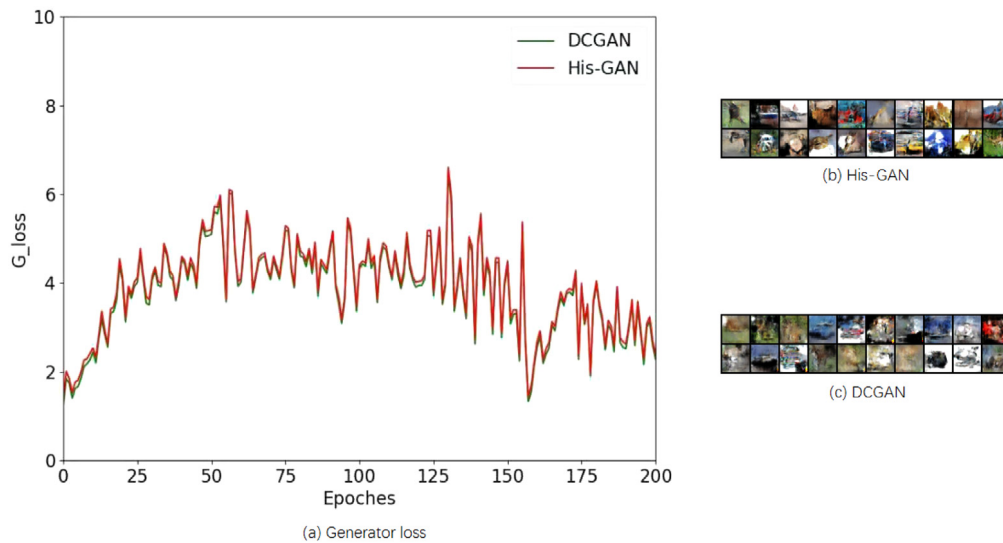


Fig. 11. Sub-figure (a) indicates the generator loss for both *His-GAN* and *DCGAN*. Sub-figure (b) indicates the simulation data generated by the *His-GAN*, while sub-figure (c) indicates the simulation data generated by the *DCGAN*.

Table 13

Recruitment counts by the number of participants who achieved these counts.

Recruit counts	Participants
0	1382
1	22
2	9
3	28
4	20
5	16
6	32
7	128
8	5
18	1

recruitment was defined as a case when a peer recruited smoker was registered on the online tobacco intervention. Further details of this study have been published in [Sadasivam, Volz, Kinney, Rao, and Houston \(2013\)](#).

The program is to specifically target these users who can recruit many additional members to participate in our program, given limited incentives. If such members can be identified, we can use their recruitment strategies to save the limited financial resources and to maximize the program's reach for these members, without wasting funds on ineffective members. [Table 13](#) describes the distribution of members recruited by recruiters.

The dataset consists of 1643 users with 68 features, in which 261 are active recruiters and 1382 of which did not recruit other users. This dataset has no explicit categories because it is a questionnaire investigation. We adopt the Min–Max normalization method to normalize all samples before applying GAN model to this dataset, and the hyperparameters are shown in the right part of [Fig. 7](#).

We randomly run our model 3 times, with three gradient methods (Adam [Kingma & Ba, 2014](#), Stochastic Gradient Descent (SGD) [Saad, 1998](#) and Nesterov [Nesterov et al., 2007](#)). Each run produces one generated dataset with the same size (1640). The statistical results are shown in [Fig. 12](#).

Although the Smoking Cessation dataset holds 68 features, all values lie in the interval $([0, 1])$ after normalization. Since we set the number of sub-interval as 10, thus, we use an array with size of 10 to save those values. The array[0] takes up the values $\in [0, 0.1)$, and so on. We count each entry and plot the array using histogram, which is shown in [Fig. 12](#). In [Fig. 12](#), the

Table 14

The results of generated data quality assessment by using Hellinger distance, KL divergence, JS divergence, Wasserstein distance and Histogram Intersection Kernel.

ID	HD	KL divergence	JS divergence	WD	HIK
(a)	0.2817	0.2839	0.0746	0.0481	79 837
(b)	0.1989	0.1811	0.0381	0.0343	90 100
(c)	0.1807	0.1278	0.0314	0.0214	95 388

Table 15

We use the normalization method to transform the absolute distances shown in [Table 14](#) into relative distances. Here the maximum value for HIK is 111 520 (1640 * 68).

HD	KL divergence	JS divergence	WD	HIK
0.7061	0.6379	0.5107	0.7131	0.3239
0.6415	0.4502	0.4209	0.4449	0.4908

blue histogram indicates the statistical count of original data at each sub-interval, while the orange histogram indicates that of generated data. Intuitively, those generated data indicated by the sub-figure (c) is the best, because the statistical results of original data are similar to that of generated data at each sub-interval. We then measure the similarity between generated data and original data reflected by [Fig. 12](#) using Hellinger distance, KL divergence ($KL(P_{data}(x) \parallel p_z(z))$), JS divergence ($0.5 * P_{data}(x) \parallel p_z(z) + 0.5 * (P_z(z) \parallel p_{data}(x))$), Wasserstein distance ($WD(P_{data}(x), p_z(z))$) and Histogram Intersection Kernel ($HIK_{\cap}(P_{data}(x), p_z(z))$). The results are shown in [Table 14](#).

In Smoking Cessation dataset, each sub-interval does not equal to 0, which means that each entry within probability percentile for both original data and generated data do not equal to 0. Thus, the scenario of $\log(0)$ does not exist. The KL divergence avoids the value of inf . Since those results belong to absolute distances, we continue to use the normalization to transform the results shown in [Table 14](#) into relative distances, and the results are shown in [Table 15](#).

The results in [Table 15](#) show the effectiveness of histogram-based measurement in numerical dataset. In other words, histogram can measure similarity between original data and simulation data as long as we map the two datasets into histogram, and no matter what generated data are and no matter how the quality of generation is. We continue to integrate histogram-based measurement score into training process of GAN

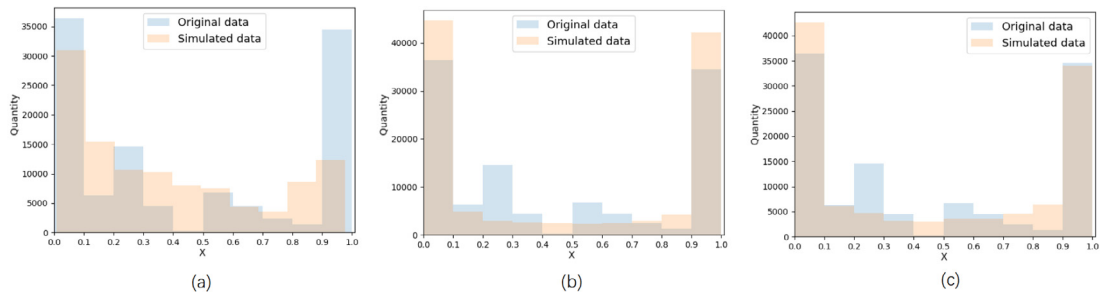


Fig. 12. The statistical results of generated data on smoking cessation dataset. There are three generated datasets, which are generated by regular GAN model with Adam (Kingma & Ba, 2014), SGD (Saad, 1998) and Nesterov (Nesterov et al., 2007) respectively. Here we set the number of sub-interval as 10, and we use an array with size of 10 to save the values. Each entry within the array indicates a sub-interval. We map all values into the interval and count each entry in terms of values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 16

We map generated data shown in Fig. 13 and sub-figure (c) of Fig. 12 into histogram, and measure their similarity with original data. The score shows that our approach is effective.

	HD	KL divergence	JS divergence	HIK
GAN	0.1807	0.1278	0.0314	95 388
His-GAN	0.1802	0.1269	0.03015	96 629

model shown in Algorithm 1 ($Epoch = 60$). We adopt the same architecture as the right part of Fig. 7, and we still adopt JS-divergence to calculate the measurement score. The generated data are shown in Fig. 13. In Fig. 13, sub-figure (a) shows loss values of two generators from His-GAN and regular GAN while sub-figure (b) shows the histograms of generated data from His-GAN and original data samples, respectively. Trend of generator loss in Smoking Cessation dataset is similar to other two datasets. His-GAN's generator loss is larger than GAN's generator loss in early training. When training goes on, generated data gradually resemble original data, and the measurement score becomes smaller. In this way, the His-GAN's generator loss would approach to GAN's generator loss. By comparing, the generated data quality produced by His-GAN (sub-figure (b)) is better than traditional GAN model (sub-figure (c) of Fig. 12). We also measure histogram similarity (bins = 10) with f-divergence and Histogram Intersection Kernel (HIK), and results are shown in Table 16.

5.4. Discussion

Results (from Tables 3 to 15) show the effectiveness of histogram on measuring the similarity between generated data and original data. In this way, we can help GAN model generate high generated data quality by integrating the histogram-based measurement score into training of GAN model. During calculating the similarity, although the five metrics (HD, KL divergence, JS divergence, WD and HIK) show similar trend on assessing generated data quality, each individual holds its own characteristics in histogram.

Although KL divergence produces results in Smoking Cessation dataset, KL divergence is not, theoretically, used for calculating similar distance measure (because of asymmetry). Assuming original data distribution follows $P(X)$, and two generated data distribution are $Q(Z_1)$ and $Q(Z_2)$ respectively. The purpose of KL divergence is to measure information loss when distribution $Q(Z_i)$ ($i = 1, 2$) is used to approximate original data distribution $P(X)$. In the sub-figure (c) of Fig. 12, statistical distribution of generated data is similar to original data, which means that both of them have similar amount of information when we apply KL divergence to calculate measurement. The smaller the KL divergence is, the smaller the information loss becomes. The two datasets have the same information, which means that there is no information

loss and KL divergence is 0. Thus, KL divergence is used for measuring similarity between two distributions via information loss. As to image dataset (e.g., Tables 3, 8), an image always contains rich information while GAN model cannot control what kind of simulation data would be generated (e.g., 'King' – 'Man' + 'Woman' = 'Queen' (Alec Radford & Chintala, 2015)). Thus, generated image could contain more or less information comparing with original image. If information contained by the generated image is not enough, $P_z(z_i)$ would be 0 (which leads to $\log(0)$), and KL divergence fails to determine which generated data is similar to original data.

JS divergence is based on KL divergence, and the difference is the denominator where the term is summation of both $P_{data}(x)$ and $P_G(x)$. JS is symmetric and it is always a finite value, while its disadvantage is that JS is useless when overlapping area between distribution of original data and that of generated data has a neglected area or even none. However, it can avoid such scenario when we map the two datasets into histogram space instead of distribution space. It can calculate similarity even though generations are noisy. Also, after normalization, JS divergence has better evaluation than other three metrics of f-divergence no matter what type (image or numeric dataset) a dataset is.

Wasserstein distance can still reflect dissimilarity of two distributions even though those distributions have no overlapping area. When we map generated data and original data into histogram, Wasserstein distance displays similarity trend as JS divergence (See Tables 4, 6, 9, 11 and 15). In comparison with JS divergence, we recommend JS divergence in histogram space, given that distance of JS divergence is smaller than that of Wasserstein distance in most cases.

As to Hellinger distance, it can directly quantify similarity between two datasets via probability distribution (or probability percentile value), and it basically focuses on probability of each sub-interval. Although Hellinger distance reflects similarity trend (the clearer generated data, the smaller the score), the scores are still relatively larger than other three metrics (KL, JS and Wasserstein distance) after normalization. Thus, Hellinger distance may not be a good choice in histogram space.

Histogram Intersection Kernel has been effective for image classification (Barla et al., 2003), our study applies histogram to evaluate generated data quality. In fact, no matter how complicated generated data are and no matter what quality of generated data is, it always holds a certain value. Histogram is an effective representation for this value, and it measures degree of similarity between two datasets by counting probability percentile value in each bin. Moreover, GAN model specializes in generating realistic simulation data, and histogram of generated data becomes more similar to that of original data as training continues (See Figs. 8 and 12). Thus, histogram intersection kernel is also a good choice.

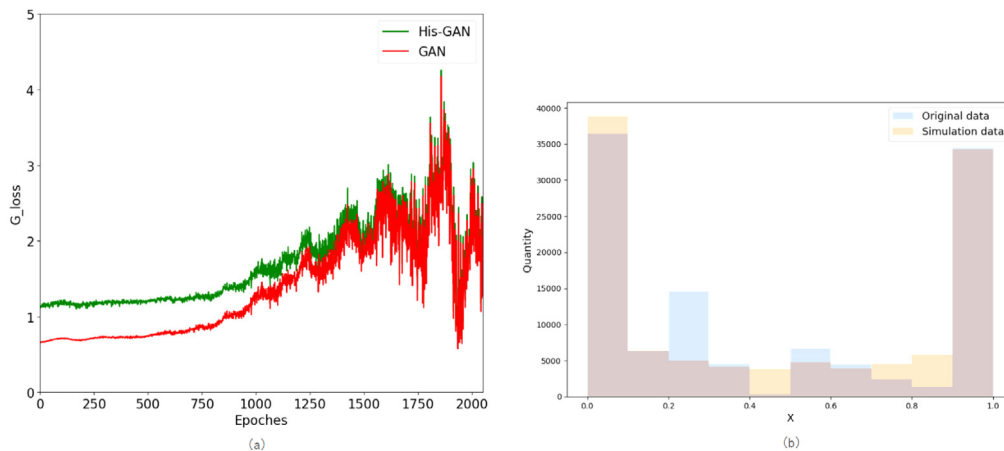


Fig. 13. Sub-figure (a) indicates the generator loss for both *His-GAN* and regular GAN. Sub-figure (b) indicates the histograms of original data and simulation data generated by the *His-GAN*.

His-GAN model is based on objective measurement score. In *His-GAN* model, we expand this score into original generator loss, given that loss would influence parameters updating of generator. If discriminator reaches into the optimal status ($D(x) = 1$ and $D(G(z)) = 0$), generator loss would be 0. However, the score can help update the parameters of generator. If GAN model has been trained successfully, *His-GAN* would converge to a point where $D(x) \approx D(G(z)) \approx 0.5$, because generator has produced realistic data and measurement score is very small. Both Figs. 9 and 11 have shown such a trend.

6. Conclusion

In this paper, we explore to improve the quality of generated data produced by GAN model. Specifically, we map the generated data and original data into a single histogram, and use two objective evaluation strategies, f-divergence community and Histogram Intersection Kernel, to calculate relative frequency statistics on each sub-interval to obtain the similarity between generated data and original data. After that, we propose *His-GAN*, which is to incorporate histogram-based measurement score into training of GAN model to update generator's parameters with back-propagation. Also, we adopt group-based method to provide a trade-off between a stable model and generated data quality. We conducted extensive experiments with MNIST and CIFAR-10 datasets, and a real-world smoking cessation dataset, to validate our idea. The results show that our approach is effective.

Acknowledgment

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

References

Alec Radford, Luke Metz, & Chintala, Soumith (2015). *Unsupervised representation learning with deep convolutional generative adversarial networks*.
 Arjovsky, Martin, Chintala, Soumith, & Bottou, Léon (2017). Wasserstein gan, arXiv preprint arXiv:1701.07875.
 Barla, Annalisa, Odone, Francesca, & Verri, Alessandro (2003). Histogram intersection kernel for image classification. In *Image processing, 2003. ICIP 2003. proceedings. 2003 international conference on, Vol. 3* (pp. III–513). IEEE.
 Chen, Xi, Duan, Yan, Houthoof, Rein, Schulman, John, Sutskever, Ilya, & Abbeel, Pieter (2016). InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets, CoRR abs/1606.03657.
 Creswell, Antonia, White, Tom, Dumoulin, Vincent, Arulkumaran, Kai, Sengupta, Biswa, & Bharath, Anil A (2018). Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1), 53–65.

Daskalakis, Constantinos, Goldberg, Paul W., & Papadimitriou, Christos H. (2009). The complexity of computing a nash equilibrium. ACM.
 Dobrushin, Roland L. (1970). Prescribing a system of random variables by conditional distributions. *Theory of Probability & Its Applications*, 15(3), 458–486.
 Gerhard, Holly E., Wichmann, Felix A., & Bethge, Matthias (2013). How sensitive is the human visual system to the local statistics of natural images? *PLoS Computational Biology*, 9,1(2013-1-24), 9(1), e1002873.
 Gretton, Arthur, Borgwardt, Karsten M., Rasch, Malte J., Schölkopf, Bernhard, & Smola, Alexander (2012). A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, 13(1), 723–773.
 Gulrajani, Ishaan, Ahmed, Faruk, Arjovsky, Martin, Dumoulin, Vincent, & Courville, Aaron C (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems* (pp. 5767–5777).
 Gurumurthy, Swaminathan, Sarvadevabhatla, Ravi Kiran, & Babu, R Venkatesh (2017). Deligan: Generative adversarial networks for diverse and limited data.. In *CVPR* (pp. 4941–4949).
 Han, Jun, & Moraga, Claudio (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International Workshop on Artificial Neural Networks* (pp. 195–201). Springer.
 Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, & Efros, Alexei A (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1125–1134).
 Jaynes, Edwin T. (2003). *Probability theory: The logic of science*. Cambridge university press.
 Joyce, James M. (2011). Kullback-leibler divergence. In *International Encyclopedia of Statistical Science* (pp. 720–722). Springer.
 Kingma, Diederik P., & Ba, Jimmy (2014). Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.
 Kroese, Dirk P, Rubinstein, Reuven Y, Cohen, Izack, Porotsky, Sergey, & Taimre, Thomas (2013). Cross-entropy method. In *Encyclopedia of Operations Research and Management Science* (pp. 326–333). Springer.
 Li, Chaozhuo, Wang, Senzhang, Yu, Philip S, Zheng, Lei, Zhang, Xiaoming, Li, Zhoujun, et al. (2018). Distribution distance minimization for unsupervised user identity linkage. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (pp. 447–456). ACM.
 Lopezpaz, David, & Oquab, Maxime (2016). Revisiting classifier two-sample tests for gan evaluation and causal discovery.
 Manning, Christopher D., & Schütze, Hinrich (1999). *Foundations of statistical natural language processing*. MIT press.
 Mao, Xudong, Li, Qing, Xie, Haoran, Lau, Raymond YK, Wang, Zhen, & Smolley, Stephen Paul (2017). Least squares generative adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on* (pp. 2813–2821). IEEE.
 Mehdi Mirza, Simon Osindero (2014). *Conditional generative adversarial nets* (pp. 2672–2680).
 Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, Bengio, Yoshua, et al. (2014). Generative adversarial nets. In *NIPS*.
 Nesterov, Yurii, et al. (2007). *Gradient methods for minimizing composite objective function*. Citeseer.
 Nowozin, Sebastian, Cseke, Botond, & Tomioka, Ryota (2016). F-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems* (pp. 271–279).
 Pathak, Deepak, Krähenbühl, Philipp, Donahue, Jeff, Darrell, Trevor, & Efros, Alexei A. (2016). Context Encoders: Feature Learning by Inpainting, CoRR abs/1604.07379.

- Pizer, Stephen M, Amburn, E Philip, Austin, John D, Cromartie, Robert, Geselowitz, Ari, Greer, Trey, et al. (1987). Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3), 355–368.
- Rebecca Kinney, L., Sowmya Rao, R., Sadasivam, R. S., Erik Volz, M., & Thomas Houston, K. (2013). *Share2quit: Web-based peer-driven referrals for smoking cessation*, vol. 2. JMIR Res Protoc, e37.
- Saad, David (1998). Online algorithms and stochastic approximations. *Online Learning*, 5.
- Sadasivam, R. S., Cutrona, S. L., Luger, T. M., Volz, E., Kinney, R., Rao, S. R., et al. (2016). Share2quit: Online social network peer marketing of tobacco cessation systems. *Nicotine & Tobacco Research*.
- Sadasivam, R. S., Volz, M. Erik, Kinney, L. Rebecca, Rao, R. Sowmya, & Houston, K. Thomas (2013). Share2quit: Web-based peer-driven referrals for smoking cessation. *JMIR Research Protocols*, 2(2), e37. <http://dx.doi.org/10.2196/resprot.2786>.
- Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, & Chen, Xi (2016). *Improved techniques for training gans*.
- Simpson, Douglas G. (1987). Minimum hellinger distance estimation for the analysis of count data. *Journal of the American statistical Association*, 82(399), 802–807.
- Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, & Salakhutdinov, Ruslan (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1), 1929–1958.
- Theis, Lucas, Oord, Aäron Van Den, & Bethge, Matthias (2015). A note on the evaluation of generative models. *Computer Science*.
- Wolfensberger, Wolf P, Nirje, Bengt, Olshansky, Simon, Perske, Robert, & Roos, Philip (1972). *The principle of normalization in human services*. National Institute on Mental Retardation.
- Wu, Yuhuai, Burda, Yuri, Salakhutdinov, Ruslan, & Grosse, Roger (2016). On the quantitative analysis of decoder-based generative models, arXiv preprint arXiv:1611.04273.
- Xu, Bing, Wang, Naiyan, Chen, Tianqi, & Li, Mu (2015). Empirical evaluation of rectified activations in convolutional network. *Computer Science*.
- Yang, Shan, Xie, Lei, Chen, Xiao, Lou, Xiaoyan, Zhu, Xuan, Huang, Dongyan, et al. (2017). Statistical parametric speech synthesis using generative adversarial networks under a multi-task learning framework, CoRR abs/1707.01670.
- Zhang, Yuxiang, Fu, Jiamei, She, Dongyu, Zhang, Ying, Wang, Senzhang, & Yang, Jufeng (2018). Text emotion distribution learning via multi-task convolutional neural network. In *IJCAI* (pp. 4595–4601).
- Zhang, Xiaoming, Hu, Xia, Wang, Senzhang, Yang, Yang, Li, Zhoujun, & Zhou, Jianshe (2016). Learning geographical hierarchy features via a compositional model. *IEEE Transactions on Multimedia*, 18(9), 1855–1868.
- Zhu, Shizhan, Fidler, Sanja, Urtasun, Raquel, Lin, Dahua, & Loy, Chen Change (2017). Be your own prada: Fashion synthesis with structural coherence, arXiv preprint arXiv:1710.07346.