Neurocomputing 431 (2021) 148–162

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

JDGAN: Enhancing generator on extremely limited data via joint distribution

Wei Li [e], Linchuan Xu [b], Zhixuan Liang [b], Senzhang Wang [c], Jiannong Cao [b], Thomas C. Lam [d], Xiaohui Cui [a,*]

[a] School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei, PR China
[b] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, PR China
[c] Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, PR China
[d] Laboratory of Experimental Optometry, Centre for Myopia Research, School of Optometry, The Hong Kong Polytechnic University, Hong Kong. Centre for Eye and Vision Research, Hong Kong, PR China
[e] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, Jiangsu, PR China. Jiangsu Key Laboratory of Media Design and Software Technology, Wuxi, Jiangsu, PR China. Science Center for Future Foods, Jiangnan University, Wuxi, Jiangsu, PR China

ARTICLE INFO

Article history:
Received 6 August 2019
Revised 16 May 2020
Accepted 6 December 2020
Available online 16 December 2020
Communicated by Zidong Wang

Keywords:
Mode collapse
Joint distribution
Reparameterization
GAN

ABSTRACT

Generative Adversarial Network (GAN) is a thriving generative model and considerable efforts have been made to enhance the generation capabilities via designing a different adversarial framework of GAN (e.g., the discriminator and the generator) or redesigning the penalty function. Although existing models have been demonstrated to be very effective, their generation capabilities have limitations. Existing GAN variants either result in identical generated instances or generate simulation data with low quality when the training data are diverse and extremely limited (a dataset consists of a set of classes but each class holds several or even one single sample) or extremely imbalanced (a category holds a set of samples and other categories hold one single sample). In this paper, we present an innovative approach to tackle this issue, which jointly employs joint distribution and reparameterization method to reparameterize the randomized space as a mixture model and learn the parameters of this mixture model along with that of GAN. In this way, we term our approach *Joint Distribution GAN* (JDGAN). In our work, we show that the JDGAN can not only generate high quality simulation data with diversity, but also increase the overlapping area between the generating distribution and the raw data distribution. We proceed to conduct extensive experiments, utilizing MNIST, CIFAR10 and Mass Spectrometry datasets, all using extremely limited amounts of data, to demonstrate the significant performance of JDGAN in both achieving the smallest *Fréchet Inception Distance* (FID) score and producing diverse generated data.

1. Introduction

Generative Adversarial Network (GAN) [15] has been demonstrated as the state-of-the-art generative model in various tasks of generating synthetic but realistic-like data [9,12,44,26,46,50]. A typical GAN model consists of two components, a generator and a discriminator. We usually view the generator as the forger who specializes in generating plausible data to fool the discriminator into accepting it as real data, while the discriminator could be regarded as a detective who can determine whether the current data are from the generator or the real dataset. The GAN model sidesteps the difficulty of approximating many intractable probabilistic computations, for it samples data from an easy-to-sample distribution so the Markov chains [36] are never needed. Its gradients are tuned by using back-propagation, which makes the training computationally inexpensive. Besides the elegant framework, the GAN model also enjoys the power of deep generative neural networks, which, in theory, have the capabilities of approximating any complicated probability distributions with adequate data. In particular, the generator is a deep neural network, which is designed to transform a noise sampled from a fixed, easy-to-sample distribution (e.g., Uniform distribution with $(-1, 1)$) into the "realistic" data. Because of its huge potentials, it has become an active research field and many researchers try to enhance data-generation capability by modifying the framework and its two components or loss function (e.g., Wasserstein GAN (WGAN) [4], Least Squares GAN (LSGAN) [34], Mixture Generators GAN (MGAN) [20] and Relativistic average GAN (RaGAN) [22]). Although

* Corresponding author.
E-mail addresses: cs_weili@jiangnan.edu.cn (W. Li), linch.xu@polyu.edu.hk (L. Xu), zhixuan.liang@connect.polyu.edu.hk (Z. Liang), szwang@nuaa.edu.cn (S. Wang), csjcao@comp.polyu.edu.hk (J. Cao), thomas.c.lam@polyu.edu.hk (T.C. Lam), xcui@whu.edu.cn (X. Cui).

https://doi.org/10.1016/j.neucom.2020.12.001
0925-2312/

those GAN variants have been widely used in many applications [26,31,11,21] and achieve impressively plausible simulated data, their generation capabilities are not viable as the training data samples are extremely limited and diverse, e.g., a dataset consists of a set of classes but each class just holds several or even one single sample. If the training dataset is extremely limited dataset, the generated instances either hold identical samples or low quality.

In deep learning, the overlapping area refers to such an area which is intersected by two distributions. Fig. 1 shows a less formal but more pedagogical illustration for overlapping area. In Fig. 1, there exists two distributions, which are generated distribution and original data distribution. We also assume both of two distributions lying in a mapping space. From Fig. 1, we can see that the two distributions just hold a small overlapping area in this mapping space, which is indicated by gray shade region. Diversity problem is also termed as mode collapse problem, which indicates generator holding a part of modes of data distribution and missing other modes. In addition, the generated data contain identical instances. In general, the Jensen Shannon Divergence is maxed out when the generated and real data distributions have disjoint support [4]. Under such a scenario, the generated data distribution $p_G(z)$ is far away from the original data distribution $p_{data}(x)$ in the mapping space, which results in that the generator fails to correctly capture all the modes of the data distribution [23]. This becomes more challenging if the training data are extremely limited data samples.

To visually and effectively demonstrate our hypothesis, we apply recent GAN variants to a diverse and extremely limited training dataset. Here, we take the MNIST dataset as the example to form the extremely limited dataset. Only 10 samples are selected from MNIST (figures '0' to '9'), and each sample belongs to a specific category. They are diverse because each sample belongs to one category; they are extremely limited because each category only holds one sample. The generated images are shown in Fig. 2.

From Fig. 2, we can see that the mode collapse arises in most existing GAN variants, i.e., most generated images are identical or low quality. It is noticed that there are five GAN variants (sub-figures (a)–(d) and (f)) sample noise from Gaussian (0, 1). DeLiGAN (sub-figure (e)) [17] samples noise from the Mixture-of-Gaussians model [47]. In addition, MGAN employs 10 generators and each generator produces a single image. Also, we demonstrate the performance of those GAN variants in imbalanced dataset (a category holds a set of samples while other categories just hold a single sample), and we can observe the details in experiment section.

Wasserstein GAN (WGAN) [4] utilizes Wasserstein distance to measure the dissimilarity between original data distribution p_r and generated data distribution p_G. However, only certain optimizers (e.g.,RMSProp [41] or SGD [6]) are suitable for optimizing WGAN. Other momentum based ones (e.g., Adam [24]) may even turn the gradients negative, causing unstable training. RaGAN [22] argues that the minimax game should simultaneously decrease the probability that real data is real. The researchers [22] induce this property by using a "relativistic discriminator", and prove that the relativistic discriminator makes training more stable. However, it is not suitable for extremely limited training data. MGAN [20], on the other hand, employs a mixture of generators to learn disconnected manifolds. Since there are no restrictions enforcing generators to learn those manifolds mutually exclusively, generators may learn the same manifolds as each other, resulting in the generation of identical instances.

When training a GAN, we usually learn a mapping from noise distribution $p_z(z)$ to original data distribution $p_r(x)$. To guarantee learning successfully, it requires a lot of training samples such that the generator can disentangle the underlying factors of variation and make the generated data diverse. When training data are extremely limited, such a mapping mechanism becomes infeasible. To address this issue, this paper proposes to explore an alternative direction, which increases the power of the latent distribution. This is because $p_z(z)$ is contained in $p_G(z)$ [3].

To this end, we propose **Joint Distribution GAN (JDGAN)**, which jointly employs multiple easy-to-sample distributions to construct the randomized space Z and learns these distributions together with the generator with reparameterization, without modifying the framework. JDGAN addresses the mode collapse by increasing the dimension of $p_z(z)$ to increase that of $p_G(z)$, because $p_z(z)$ is contained in $p_G(z)$ [3]. If the supports of $p_{data}(x)$ and $p_G(z)$ are not disjoint, the generator can correctly capture all the modes. In the training of JDGAN, the overlapping area between $p_{data}(x)$ and $p_G(z)$ is significantly increased, even at the beginning of the training. To obtain a sample from the joint distribution, we employ the reparameterization method introduced by Kingma [25] to sample noise. Assuming the joint distribution is formed by a Gaussian distribution ($G(\mu, \theta)$) and a Uniform distribution ($U(a, b)$), we represent the noise from the joint distribution as a deterministic function of μ, θ, a and b.

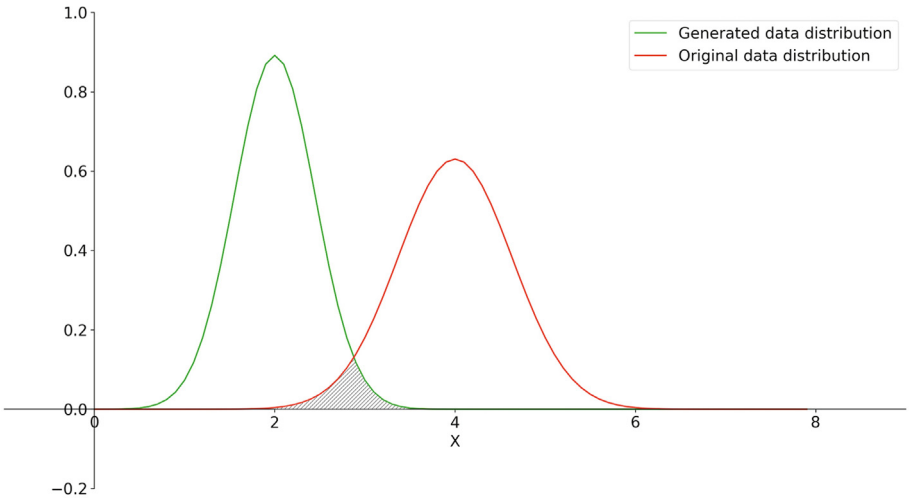


Fig. 1. The gray shade region indicates the overlapping area between two distributions.

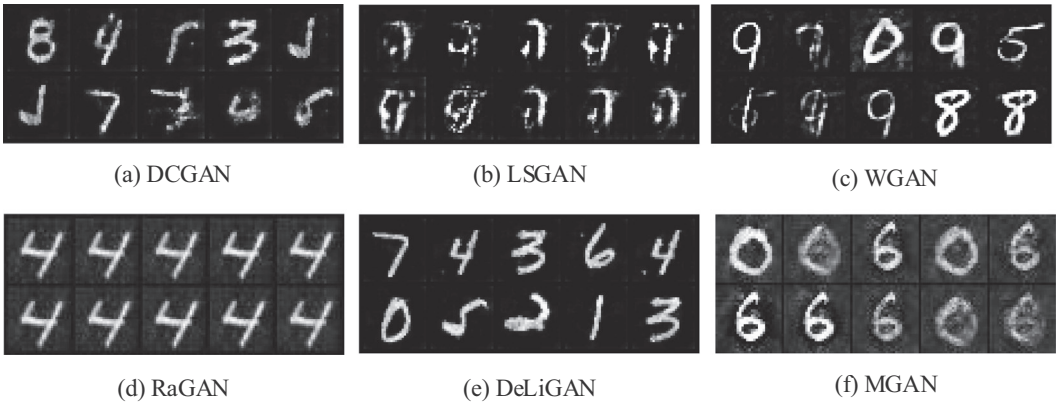


Fig. 2. A case study on extremely limited MNIST samples. The generator samples noise from standard Gaussian distribution (0, 1) for DCGAN, LSGAN, WGAN, RaGAN and MGAN, and samples noise from a Mixture-of-Gaussians model for DeLiGAN.

Since DeLiGAN [17] also modifies the input lalent space Z, a likely concern is of the difference between JDGAN and DeLiGAN. Here, we give the **major differences between our proposed JDGAN and DeLiGAN** as follows:

- The randomized space is determined by the MoG model in DeLiGAN, and MoG model is a linear combination of multiple Gaussian distributions, which causes each Gaussian distribution having different weights. In this way, the MoG model has been easily driven into the local optimal defect [7]. In other words, holding most contribution for a specific Gaussian distribution to MoG model dominates other Gaussian models. The MoG model becomes a single Gaussian model under such a scenario. Therefore, the mode collapse arises. For JDGAN, the randomized space is determined by the joint distribution and joint distribution is a non-linear distribution, which increases the power of prior distribution and can avoid $p_z(z)$ becomes a single distribution.
- When we sample the noise from the MoG model, it needs two steps. First, one of N Gaussian models is selected. Second, we draw the noise from the chosen Gaussian distribution. Considering an extreme scenario, the same Gaussian distribution may be chosen at each epoch. The MoG model becomes the single Gaussian model under such a scenario, and the mode collapse still arises.
- In our study, we fully discuss how to determine the number of distributions. However, DeLiGAN does not give how to determine the number of Gaussian model. We believe that the number of distributions would influence the diversity of generated data. More details are shown in experiment section.

In summary, the major innovations and contributions of this paper are as follows:

- This paper proposes a novel JDGAN model which modifies the input randomized space to enhance the generation capabilities of the GAN model, overcoming the problem of mode collapse.
- This paper demonstrates how to sample noise from the joint distribution and how to learn the parameters of the joint distribution together with the generator from theoretical and empirical perspectives, giving new insights into the success of JDGAN.
- Through comprehensive experiments on generating simulated data, we demonstrate the effectiveness of the proposed approach.

This paper is organized as follows. In Section 2 we discuss some related work. We discuss the limitations of a single distribution in Section 3 and present our main idea in Section 4. In Section 5 we will show our experimental results, and we conclude this work in Section 6.

2. Related work

The GAN model displays its powerful generative capabilities since its invention, however, the limitations (e.g., the loss of diversity, mode collapse and vanishing gradients) are also obvious. Hence, many modifications to the original GAN model have been proposed, and they can be mainly categorized into three types: modifying the components, modifying the penalty function and modifying the architecture.

Modifying the components. The early researchers utilize some seemingly simple but powerful litter tricks to improve the performance of the GAN model. One of the first major improvements is DCGAN [1], which hopes to bridge the gap between the success of CNNs for supervised learning and unsupervised learning. It modifies the two components of the GAN model by adding some tricks such as BatchNorm [30], ReLu activation [14] for the generator and BatchNorm, LeakyReLu activation [45] for the discriminator, and replacing any pooling layers [27] with strided convolutions (discriminator) and fractional-strided convolutions (generator). These modifications are suitable for both the generator and the discriminator to learn good up-sampling and down-sampling operations, which could improve the quality of simulated data. Salimans et al. (Improved GAN) [39] propose heuristic approaches to stabilize the training of GAN. Specifically, they use the feature matching to address the instability of training a GAN model by changing the objective for the generator to prevent the problem of overtraining, and they use the mini-batch [30] discriminator to prevent the generator collapse, for the discriminator can easily tell whether the generator is producing same outputs. Mirza et al. (Conditional GAN) [35] train both the discriminator and the generator by using the new input that is conditioned on adding extra information y (e.g., class labels), and it can generate descriptive tags which are not part of training labels. This modification holds considerable flexibility for generation.

Modifying the penalty function. The GAN model generally adopts the JS divergence [33] to calculate the similarity between two distributions which are from different datasets, for the principle of GAN is to transform a distribution into another distribution. However, it is hard to achieve the transforming process, and always causes the mode collapse in practice. Thus, many researchers adopt different strategies to address this issue. WGAN [4] replaces the JS divergence with the Wasserstein distance [37]. The value of JS divergence for a GAN model could be a constant

as the two distributions have no overlapping area (or the overlapping area can be neglected), which would cause the vanishing gradients. The Wasserstein distance can reflect the dissimilarity between the two distributions without an overlapping area. Moreover, Gulrajani et al. (Improved WGAN) [16] find that the weight clipping adversely reduces the capability of the discriminator in WGAN, they then improve the WGAN by penalizing the norm of the discriminator gradients during training instead of performing parameter clipping. LSGAN [34] modifies the loss function with the Least Squares to generate samples that are closer to the real data. This study argues that the Sigmoid Cross Entropy loss function [49] for the discriminator would lead to the problem of vanishing gradients as updating the generator using the fake samples that are on the correct side of the decision boundary and are far from the real data. The Least Squares loss function can penalize the fake samples that are lying in a long way on the correct side of the decision boundary moving toward the decision boundary even though they are classified correctly.

Modifying the Architecture. Since manifolds of original data are disconnected in the space and a single generator G only produces instances in certain regions of this space, the generated data focuses on several or one single manifold. Researchers attempt to increase the quantity of generator to learn more about different manifolds. Tolstikhin et al. [42] added a new component to a mixture model by running a GAN algorithm on a re-weighted sample. Inspired by boosting techniques, this idea greedily aggregates many potentially weak individual predictors to form a strong composite predictor. This model is termed AdaGAN. Since AdaGAN utilizes a sequential training technique to train the model, the model is computationally expensive. Moreover, it is hard to search the *ChooseMixtureWeight* and the *UpdateTrainingWeight* functions for boosting techniques. Arora et al., [5], alternatively, trained a mixture of generators and discriminators to play the minimax game with the reward function being the weighted average reward function between any pair of generator and discriminator. This strategy is not only computationally expensive but also lacks a mechanism to enforce the divergence among generators. Ghosh et al. [13] employed many generators and trained them by using multi-class discriminators that, in addition to detecting whether a sample is fake or not, predict which generator produces this sample. The loss function in this study focuses on detecting whether a sample is fake and does not directly encourage generators to produce diverse instances. The recent GAN variant for increasing the quantity of generator is MGAN [20]. MGAN employs many generators $G_{1:K}$ and an extra classifier C to construct architecture. Although study [20] claimed such a design can help a model learn all manifolds, it is difficult to achieve this goal in practice. MGAN does not provide that G_i is mutually exclusive with $G_j, i \neq j$. In other words, G_i and G_j may learn the same manifold during training. In addition, MGAN adopts the shared parameters to save training cost. However, the shared parameters may cause all generators output the same instance, given that the parameters have an influence to the quality of data. RaGAN [22] argues that the Nash equilibrium should simultaneously decrease the probability that real data is real, which may improve the GAN performance. In this way, RaGAN utilizes a "relativistic discriminator" to prove this property, and this discriminator estimates the probability that the given real data is more realistic than fake data, on average. However, it still suffers from the challenge in the case of extremely limited data. We can observe more details in section experiments.

Modifying randomized space. DeLiGAN [17] hopes to make $\mathscr{Z}$ disconnected to learn the disconnected manifolds, with the mixture of Gaussian (MoG) model. However, it is hard to make $\mathscr{Z}$ disconnected in practice. MoG model is a linear combination of multiple Gaussian distributions, and the combined distribution still belongs to a bounded continuous distribution. This is pedagogically shown in Fig. 3. There still exists the uncovered manifolds such that some modes are missed in generated instances, given that $p_z(z)$ is contained in $p_G(z)$ [3]. In addition, considering an extreme scenario, the same Gaussian distribution has been chosen at each epoch. The MoG model becomes the single Gaussian model under such a scenario. The mode collapse is still not addressed.

3. Preliminaries

In this section, we discuss the limitations of sampling noises from a fixed, easy-to-sample distribution and give the explanation of reparameterization method.

3.1. The limited overlapping area

Although the GAN model was introduced in Section 1, we formally present the GAN model as below to establish the continuity. The GAN model was introduced by Goodfellow [15] as a novel generative model to simultaneously train a generator and a discriminator by using Eq. (1).

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

where x comes from a distribution $p_r(x)$ sampled from the original dataset and z comes from a fixed, easy-to-sample distribution $p_z(z)$ (e.g., Uniform with $(-1, 1)$ or Gaussian with $(0, 1)$). The generator builds a mapping function from $p_z(z)$ to the original data space χ, and the discriminator would output a single score $\in [0, 1]$ to indicate whether the current data are from the generator or not. Generally, a small score indicates the current data generated by the generator, while a high score indicates the current data coming from the original dataset. We repeat this training process until both the discriminator and the generator reach to the Nash equilibrium [8] where $p_G = p_r = 0.5$.

Eq. (1) measures the difference between $p_r(x)$ and $p_G(z)$ by using the Jensen–Shannon (JS) divergence [10]. The closer the two distributions are, the smaller the JS divergence is. However, we cannot guarantee that the two probability distributions ($p_G(z)$ and $p_r(x)$) have a large overlapping area because $p_G(z)$ and $p_r(x)$ lie in low-dimensional manifolds [3,4]. That is to say, the two distributions are hard to overlap because they are disjoint in χ space. If the overlapping area is none or very small, the overlapping measure between $p_G(z)$ and $p_r(x)$ is 0 and the JS divergence is a constant (e.g., log2). The gradients are vanishing under such a scenario. Note that $p_G(z)$ is defined via sampling from the simple prior $p_z(z)$ [3]. This implicitly indicates that we can utilize prior $p_z(z)$ to increase the overlapping area.

3.2. The limited diversity

In general, the noise distribution $p_z(z)$ is a fixed, easy-to-sample distribution, i.e., Uniform with $(-1, 1)$ or Gaussian with $(0, 1)$. The model usually adopts independent identically distributed strategy to draw noise code z from such an distribution, and transform it into high-dimensional matrix ($G(z)$). Let G be a function composed by affine transformations and rectifiers. We get $G(z) = \mathrm{D}_n\mathrm{W}_n \ldots \mathrm{D}_1\mathrm{W}_1 z$ where W_i denotes the affine transformation and D_i indicates the rectifier. Note that the noise code z follows the distribution $p_z(z)$ and the generated data G_z follows distribution $p_G(z)$. In this way, $p_z(z)$ is contained in $p_G(z)$.

With this in place, the diversity of generated data G_z has been limited by such a low-dimensional noise code when GAN model has been trained successfully because the number of noise code z

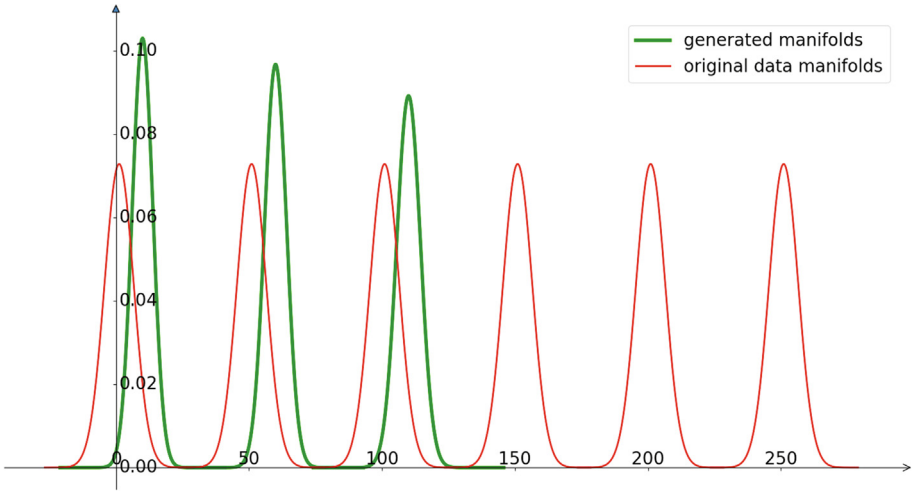


Fig. 3. Assuming original data (red curve) hold 6 submanifolds, while MoG model (green curve) just covers part of submanifolds of original data.

is limited. One may want to increase the dimensionality by increasing the number of noise samples (e.g., $z = 10 \rightarrow z = 100$). However, these noise samples could be linear correlation so the intrinsic dimensionality of those noise samples is far less than 100. It further increases the loss of diversity as we consider such a scenario where the neural network brings the dimensionality reduction mapping.

On the other hand, assuming the original data holds a set of disjoint manifolds and each one named submanifold$_i$, a part of manifolds of original data are covered by the support of the generated data distribution. This is because $p_z(z)$ is supported on a connected subspace of χ while $p_z(z)$ is contained in $p_G(z)$. Therefore, generator G (a continuous function by design) can not correctly model a set of disjoint manifolds in χ [23]. Although there is a trade-off between covering all original data distribution and minimizing the volume of the off real-manifold space in the cover, such a trade-off also indicates that the generator may sacrifice certain submanifolds to learn a cover with less off real-manifold volume [23]. In this way, the diversity of the generated data is limited.

In addition, there is a lower bound for intrinsic dimensionality of a dataset [48], the diversity of data will be lost when the intrinsic dimensionality of generated data is smaller than such a lower bound. It is much worse on diversity when the training data samples are extremely limited and diverse (See Fig. 2).

3.3. Reparameterization method

In the traditional sampling method, the lower bound of the noise variable is fixed, e.g., Uniform with $(-1, 1)$ or Gaussian with $(0, 1)$. The reparameterization method can help us to determine the lower bound of the variable through back-propagation. For example, we want to take the gradient w.r.t. θ of the following expectation $\mathbb{E}_{p(z)} f_\theta(z)$ where p denotes the density. However, computing the gradient of this expectation (i.e., $\nabla_\theta \mathbb{E}_{p_\theta(z)} f_\theta(z)$) is often difficult because the integral is typically unknown and the parameters θ, with respect to which we are computing the gradient, are of the distribution $p_\theta(z)$. The reparameterisation method can "transform" this formula to another expression, i.e., $\nabla_\theta \mathbb{E}_{p_\theta(z)} f_\theta(z) = \nabla_\theta \mathbb{E}_{p_\epsilon} f(g(\epsilon, \theta))$. In this way, the gradient is now unrelated to the distribution with which we take the expectation, so easily passes through the integral. We take the Gaussian distribution as the example. The sample can be represented from the Gaussian as a deterministic function of μ and δ and an auxiliary noise variable ϵ, i.e., $z = \mu + \delta\epsilon$, and we can learn the parameters μ and δ along with the GAN parameters.

4. Joint distribution GAN

In this section, we propose the Joint Distribution GAN (JDGAN). As mentioned in the introduction, the innovation of JDGAN lies in the input randomized space Z, which modifies Z with joint of multiple independent distributions. To this end, there are four important issues to be addressed, which are how to construct a new randomized space using joint distribution, how to draw samples from the joint distribution, how to choose the number of distributions and how to learn the parameters of JDGAN.

4.1. Constructing the new randomized space

No matter which GAN-based model we choose, we always hope the network to learn a mapping from the latent distribution $p_z(z)$ to the original data distribution $p_r(x)$. In our study, we employ joint distribution to construct the randomized space for the generator. In joint distribution, the distribution of the random vector $Z = (Z_1, Z_2, \ldots, Z_n)$ is named joint distribution of the random variable Z_i, and it is shown as follows.

$$P(Z) = P(Z_1 = z_1 \text{ and } Z_2 = z_2 \text{ and } \ldots \text{ and } Z_n = z_n) \tag{2}$$

In Eq. (4), each z_i indicates a distribution. The joint distribution of multiple variables can be generally represented as follows:

$$p_z(z) = p_z(z_1, z_2, \ldots, z_n) = p_z(z_1)p_z(z_2)\ldots p_z(z_n) \tag{3}$$

Here we assume that each distribution is independent of all the others, Eq. (3) can be reformulated as follows:

$$\begin{aligned} p_z(z) &= p(z_1, z_2, \ldots, z_N|\theta_1, \theta_2, \ldots, \theta_N) \\ &= p\left(z_1|\theta_{z_1},\ z_2|\theta_{z_2}, \ldots\ z_N|\theta_{z_N}\right) \\ &= p\left(z_1|\theta_{z_1}\right) \cdot \ldots \cdot p\left(z_i|\theta_{z_i}\right) \cdot \ldots \cdot p\left(z_n|\theta_{z_n}\right) = \prod_i^n P\left(z_i|\theta_{z_i}\right) \end{aligned} \tag{4}$$

In this way, $p_G(z)$ which is the generating distribution parameterized by the generator should be a larger and more complicated distribution, so the overlapping area between $p_r(x)$ and $p_G(z)$ is increased. Moreover, the increasing distributions also increase the dimensionality of manifold, so the diversity of the generated data can also be increased. This is because adopting the joint distri-

bution can construct a more complex $p_z(z)$. Since the supports of $p_G(z)$ and $p_r(x)$ lie on low-dimensional manifolds [3], we attempt to increase the dimension of manifold of $p_G(z)$ to overlap $p_r(x)$, given that $p_z(z)$ has to be contained in $p_G(z)$ [3]. If the dimension of manifold of $p_G(z)$ is no longer less than that of $p_r(x)$, it is certainly for $p_r(x)$ to be completely covered by $p_G(z)$ in the mapping space. Assuming an extreme scenario, $p_G(z)$ is complicated enough to fill up the mapping space. That is to say, $p_r(x)$ and $p_G(z)$ have non-disjoint supports. In this way, the diversity of the generated data can be guaranteed.

Since the JDGAN focuses on constructing the joint distribution to modify the randomized space Z, the generator and discriminator architectures of JDGAN are from the vanilla DCGAN. The details of components still adopt Conv-BatchNorm-ReLu (Generator G) and Conv-BatchNorm-LeakyReLu (Discriminator D). One may use other frameworks given that the designing of randomized space Z is independent of the architectures. Even though the common architectures are utilized, JDGAN is demonstrated to be more effective than state-of-the-art variants of GAN. More details are shown in the experiment section.

Next, we discuss how to choose each distribution (i.e., $P(z_i|\theta_{z_i})$) to form the joint distribution for constructing randomized space Z. Theoretically, we can use any statistical distribution (even the same distribution) to jointly construct Z, and what we need to pay attention to is that those distributions have to be independent to each other when employing multiple distributions (we will consider the case of dependent joint distribution in future work). In fact, many distributions can be viewed as the combination of the Gaussian distributions and the Uniform distributions [19], and both of them are widely used in many applications. Here we take the two distributions as the example to construct the randomized space. One can try to use other distributions, based on different tasks. We instantiate Eq. (4) via the joint distribution of a Gaussian distribution and a Uniform distribution as shown in the following equation:

$$p_z(z) = p(z_1, z_2) = p(z_1|G(\mu, \delta))p(z_2|U(a, b)), \tag{5}$$

By Eq. (5), the joining of two distributions belongs to non-linear combination, which increases the power of prior distribution. It is highly likely to produce a larger overlapping area than a single one and the joining of two distributions significantly increases the diversity. Next, we will introduce how to sample noise from the new randomized space.

4.2. Sampling from joint distribution

We take the Eq. (5) as the example. In Eq. (5), the new randomized space has been determined by two different distributions ($p(z_1)$ follows the Gaussian distribution and $p(z_2)$ follows the Uniform distribution). In order to draw noise code z from Eq. (5), we employ the "reparameterization method" [25] to sample noise code z_1 from the Gaussian distribution and use the inverse transformation method [43] with the cumulative distribution function ($\frac{x-a}{b-a}$) to sample noise code z_2 from the Uniform distribution. Since we repeatedly sample noise code from the two distributions and both of the two distributions are univariate Gaussian distribution and uniform distribution, all the elements of z are not the same and each element within noise code is a scalar. In this way, the noise code drawn from the joint distribution can be denoted as follows:

$$z = (\mu + \delta\varepsilon)[\zeta(b - a) + a] \tag{6}$$

In Eq. (6), the first term follows the standard Gaussian and the second term follows the standard Uniform distributions respectively, which means that the data we sampled from the randomized space Z are a deterministic function of the parameters of μ, δ, b and a. Substituting Eqs. (6) and (5), we get:

$$P_r(G(z)) = \int p_r[G(\mu + \delta\varepsilon)|\varepsilon]p(\varepsilon)d\varepsilon \cdot \int p_r[G(\zeta(b - a) + a)|\zeta]p(\zeta)d\zeta \tag{7}$$

In this way, our new objective is to learn μ, δ, b and a together with the generator to minimize $\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$, and the architecture of JDGAN is shown in Fig. 4.

4.3. Learning parameters of JDGAN

For each distribution, we first need to initialize their parameters. We still take the Eq. (6) as the example. We initialize μ_i and δ_i with 0 and 1 for $p(z_1)$ and assign (-1, 1) to $p(z_2)$ (a and b). In order to train our model, generator samples a latent vector z from the joint distribution (Eq. (6)). After feeding z into the generator, we obtain the generated data and then put these data into discriminator. The discriminator would output a signal to update generator's parameters by using regular GAN training procedure (Eq. (1)). Also, μ, δ, a and b are trained simultaneously along with the generator's parameters, with gradient methods. The derivative with respect to each parameter is easy to obtain using the back-propagation method (e.g., $\frac{\partial G}{\partial \mu} = \frac{\partial G}{\partial z} \cdot \frac{\partial z}{\partial \mu}$).

4.4. Determining the number of distributions

The goal of jointly employing multiple distributions in the randomized space is to increase the generating capabilities. However, we need to know how to choose the optimal number of multiple distributions. In this paper, the *Intrinsic Dimension* [2,29] is utilized to determine the number of distributions. A dataset is usually projected into a low-dimensional manifold, and the minimal dimension of such a manifold is called *intrinsic dimension* [2]. Assume there exist independent identically distributed (IID) sample observations $X_1, \ldots, X_n$ from a high-dimensional space $\mathbb{R}^D$, and those observations represent an embedding of lower-dimensional samples, i.e., $X_i = g(Y_i)$. Y_i is from an unknown smooth density f on $\mathbb{R}^m$ with $m < D$. The unknown term m is the expected *intrinsic dimension*. Here, we utilize widely used maximum likelihood [29] to estimate *intrinsic dimension*, which is shown in Eq. (8).

$$\widehat{m}(x) = \left[\frac{1}{n}\sum_{j=1}^{n} log\frac{x_j}{T_j(x)}\right]^{-1} \tag{8}$$

where $T_j(x)$ is the Euclidean distance from the point x to its jth nearest neighbor. More deductions are shown in [29]. We take the extremely limited MNIST as the example to demonstrate the optimal number of distributions. We feed the extremely limited training samples and generated instances from DCGAN (See Fig. 2(a)) into Eq. (8), and get $\widehat{m}(x)_{DCGAN} = 3.30$, $\widehat{m}(x)_{MNIST} = 5.79$. In this way, two distributions may be the optimal choice in our cases. A detailed discussion is shown in experiment section.

Besides, we observe in our experiments that a larger number not always brings better performance. The reason behind it could be as follows. If the number of distributions is large, it can cause that the noise distribution is more complicated than the raw data distribution. In other words, a GAN model transforms a complicated distribution into a relatively simpler distribution, and such a transformation process could abandon some valuable information for mapping the former one into the latter one, which would cause the loss of diversity.

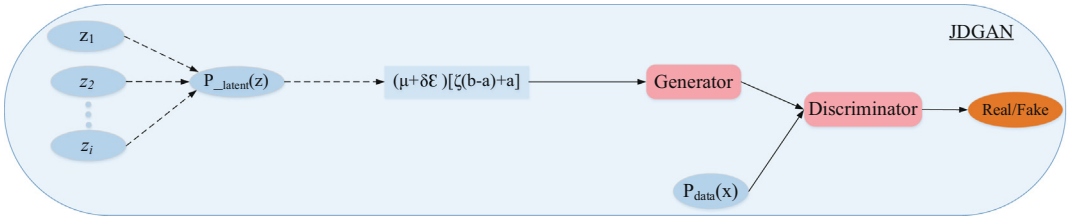


Fig. 4. JDGAN architecture. Different from the regular GAN model, the generator samples noise from the joint distribution. The $P_{latent}(z)$ is made up of multiple distributions (z_i). Those distributions are independent to each other.

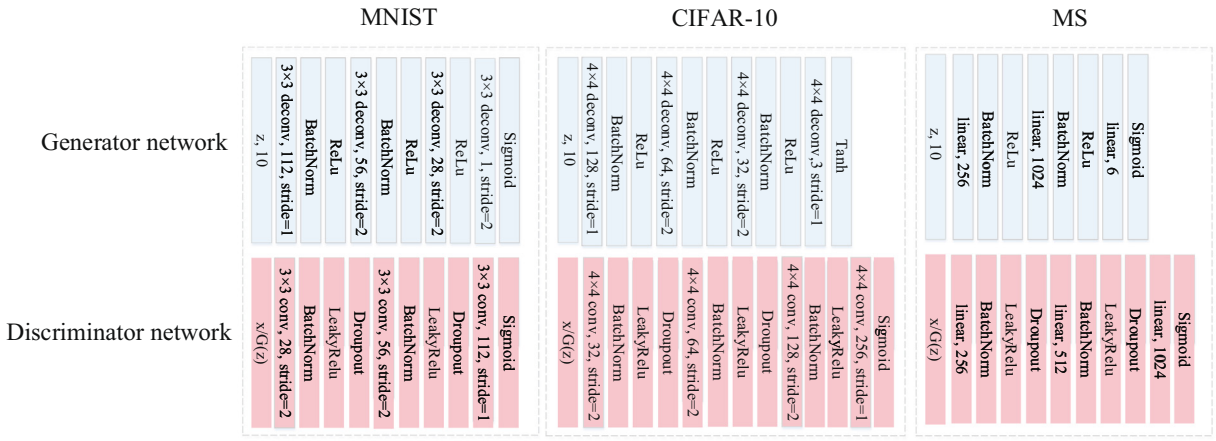


Fig. 5. Architectural details of JDGAN model. "$K \times K$, conv/deconv, C, stride = S" denotes a convolutional/deconvolutional layer with $K \times K$ kernel, C output filters and stride = S. BN indicates a batch normalization layer. 10 indicates the dimension of z. In Mass Spectrometry (MS) dataset, the out channel at the last layer for generator is set to 6, for 6 features (Acq Time, Intensity, Precursor Intensity Acquisition, Apex Time, Elution Peak Width, MS2Counts) can determine whether the eyes are diseased or not.

5. Empirical evaluation

5.1. Experiment settings

For the experiments, the implementation details of JDGAN are shown in Fig. 5. Six recent GAN models are employed as baselines, which are **DCGAN** [1], **WGAN** [4], **MGAN** [20], **LSGAN** [34], **RaGAN** [22] and **DeLiGAN** [17], respectively.

Two commonly used public image datasets, the MNIST dataset and the CIFAR-10 dataset, and a real-word medical Mass Spectrometry dataset, are studied. JDGAN is proposed to enhance the capabilities of capturing complicated data distribution. We thus mainly compare the diversity of generated images with these GAN-based models. Since deep neural networks can theoretically approximate any kind of distributions with adequate training data, we reduce the amount of data so as to test how many benefits the designed JDGAN can bring. However, it is not easy to justify how limited the data should be. In this way, we study diverse and extremely limited data which means there are many categories of images but each category holds only one image because no more data can further be reduced. To make a fair comparison, the quantity of generated data for all models is the same as that of training data (10 samples for image dataset and 5 samples for medical dataset).

5.2. MNIST dataset

We first generate image simulation data using JDGAN on MNIST dataset where each sample is a gray image with 1*28*28 size. We sample only one image from each category as the training data, and the hyperparameters are shown in Fig. 5. In Fig. 5, we use the Adam gradient method [24] and Binary Cross Entropy [28] to update the generator and the discriminator and the multiple distribution's parameters (e.g., μ, δ, b and a). We set the parameter of LeakyRelu [45] as 0.02 and that of Dropout [40] as 0.5. The activation of last layer for the generator and the discriminator is *Sigmoid* for MNIST and MS datasets, and that is *Tanh* for CIFAR-10 dataset.

We then test the number of distributions in Z from 2 to 4 and the drawing noise samples from only multiple Gaussian distributions ($z = (\mu + \delta\varepsilon_1) * \ \ldots \ * (\mu + \delta\varepsilon_n)$) and only multiple Uniform distributions ($z = (\zeta_1(b-a)+a) * \ \ldots \ * (\zeta_n(b-a)+a)$) which are used to form the joint distribution in Z, and the generated results are shown in Table 2. Note that all parameters a, b have been initialized to $(-1, 1)$ and all parameters μ and δ have been initialized to (0, 1). After that, we incorporate the two distributions (Eq. (7)) into a joint distribution, and the results are shown in Table 3. Note that we have already studied all the baselines on this dataset, and the results shown in Fig. 2 demonstrate poor performance of all the baselines. We now report the *Fréchet Inception Distance* (FID) [18,32] scores obtained by our JDGAN (Row 1 and column1 in Table 3) and baselines (Fig. 2) in Table 1 on MNIST dataset. FID measures the Fréchet distance between the two distributions and it is the 2-Wasserstein distance. The smaller the FID score is, the better performance the model holds. From Table 1, we can see that the JDGAN achieves the smallest score, which also proves that the proposed JDGAN outperforms other GAN variants.

The three sets of results demonstrate the effectiveness of our proposed JDGAN. In addition, there is an interesting scenario where the diversity is decreasing when the number of distributions is increasing. Note that when employing a large number of distributions to construct the randomized space, the noise distribution could be more complicated than original data distribution. We think it may be because the noise's diversity is richer under this scenario, and the generator may abandon some values during training when the GAN model transforms a complicated

Table 1
FID scores of generated data shown in Fig. 2 and Row 1, column1 in Table 3.

Models	FID Score
DCGAN	202.1
DeLiGAN	106.3
LSGAN	266.5
WGAN	179.4
RaGAN	290.3
MGAN	401.8
JDGAN	**66.2**

Table 2
We jointly employ multiple Gaussian distributions (Simulation Data1) or Uniform distributions (Simulation Data2). The number = 2 indicates that we employ two same distributions to construct *Z*. The best performance is achieved with the number = 2 as the randomized space is constructed by 2 Gaussian distributions, for the diversity outperforms others.

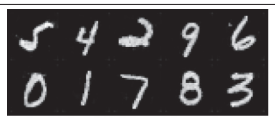
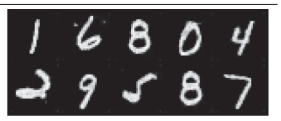
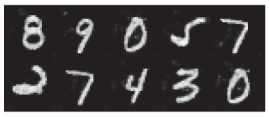
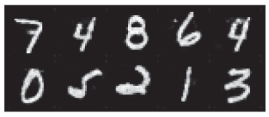
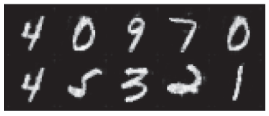
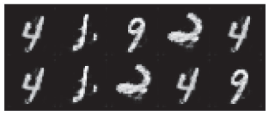
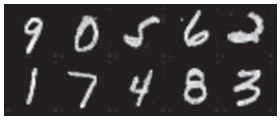
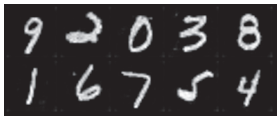
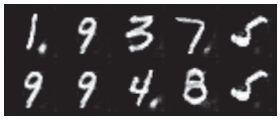
Number	Simulation Data1	Simulation Data2
2		
3		
4		

Table 3
We jointly employ multiple Uniform and Gaussian distributions. The number = 1 indicates that the joint distribution is determined by a Uniform and a Gaussian (Eq. (6)), and number = 2 indicates that the joint distribution is determined by two Uniform distributions and two Gaussian distributions. The best performance is achieved with the number = 1 and number = 2, for they show the promising diversity.

Number	Simulation Data
1	
2	
3	

distribution into an relatively simpler distribution. The abandoned values would cause incomplete information, which may be a part of a handwritten figure. Thus, to fool the discriminator, the generator easily tends to generate the "realistic" but repeated simulated data rather than the "unrealistic" but diverse instances. Thus, a suitable number is 2 for the MNIST dataset in this scenario.

Previous experiments present a visual evaluation of the generation and show that all the baselines suffer from the mode collapse problem. To further examine the advantage of our idea, we present the overlapping area between the generating distribution and the raw data distribution in Fig. 6 . It shows that JDAGN has the largest overlapping area with the raw data distribution and thus suffers the least from the mode collapse.

We also conduct experiments to validate the non-negligible overlapping area at the early step, and the result is shown in Fig. 7. Fig. 7 shows that the overlapping area between the distribution of simulated data generated by JDGAN (Eq. (6)) and that of original data is considerably large while it is not the case with all the baselines.

Notice that all the experiments shown above are conducted on the scenario with diverse and extremely data. To validate the generalization of proposed JDGAN, we test its performance on other scenarios.

1. We apply our proposed JDGAN to the adequate training data (60000 figure samples) and compare it with the state-of-the-art GAN variants. The generated images are shown in Fig. 8. It shows that the proposed JDGAN can also achieve good performance with adequate training samples.
2. The training dataset is imbalanced dataset. On this scenario, the category '0' holds 8 samples, the categories '1' and '2' hold 1 sample. The results are shown in Fig. 9. The generated data produced by JDGAN are diverse because the three categories are arisen while DeLiGAN lacks category '2'.

In this case, the experimental results show the effectiveness of JDGAN in extremely limited data. Next, we proceed to apply the JDGAN to the CIFAR10 dataset.

5.3. CIFAR-10 dataset

The CIFAR-10 dataset is a colorful image dataset with 3*32*32 size. We continue to apply JDGAN to this dataset, and the architectural details are the same as the Fig. 5 except for the last layer of the generator, which replaces *Sigmoid* with *Tanh*, given that the CIFAR-10 dataset is colorful dataset and the *Tanh* function can cover the color space of the training distribution. We set the noise distribution as the Gaussian distribution (0, 1) for DCGAN [1], WGAN [4], RaGAN [22], MGAN [20] and LSGAN [34], and sample noise samples from the Mixture-of-Gaussians model for DeLiGAN [17]. For MGAN, we employ 10 generators and each one only produces one instance. Similar to MNIST dataset, we still samples 10 images from CIFAR10 dataset, and each image represents a specific category. The generated images of different models are shown in Fig. 10. From these generated data, we can see that there are many identical images in sub-figures (a), (b), (d) and (f), and we even observe the noise generated images in sub-figures (c), (d) and (f).

Similar to MNIST, we continue to construct the randomized space *Z* by using only multiple Gaussian distributions, only multiple Uniform distributions and the joint distribution of these two distributions respectively. The generated results are shown in Tables 4 and 5, respectively.

We can see that the best performance is achieved with the number = 3 in Table 4 when the randomized space is determined by three same distributions, and that is achieved with the number = 2 in Table 5 if the randomized space is determined by two different distributions.

We think that the information possessed by CIFAR-10 dataset is richer than MNIST dataset, and there is a need to sample noise from a more complicated distribution for the generator to capture

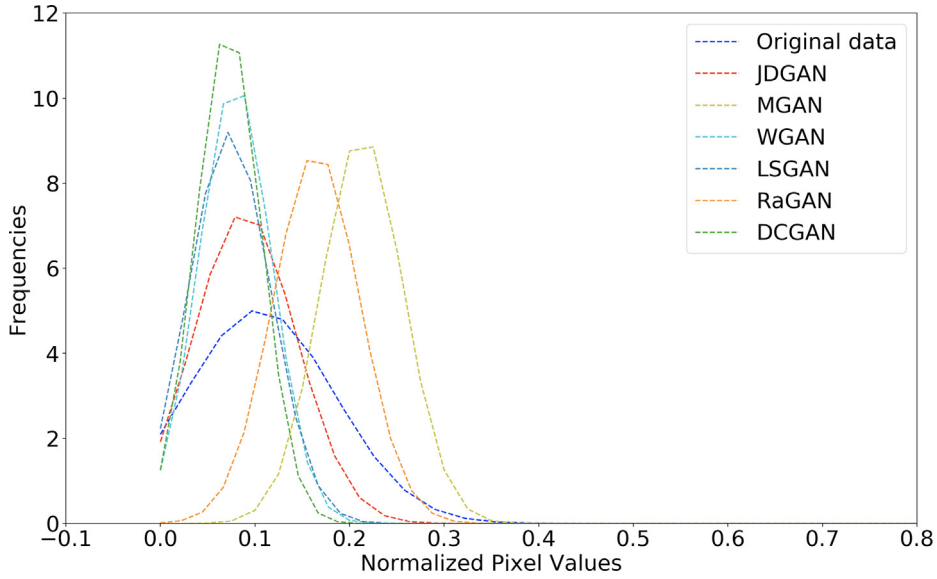


Fig. 6. The overlapping area on the MNIST dataset. The blue line indicates the original data distribution, and other lines indicate the simulated data distribution, which are generated by LSGAN, WGAN, DCGAN, RaGAN, MGAN and JDGAN respectively. JDGAN holds the largest overlapping area. Here we use the *plt.hist*() and *MLA.normpdf*() functions with *bins* = 10 to plot their histograms.

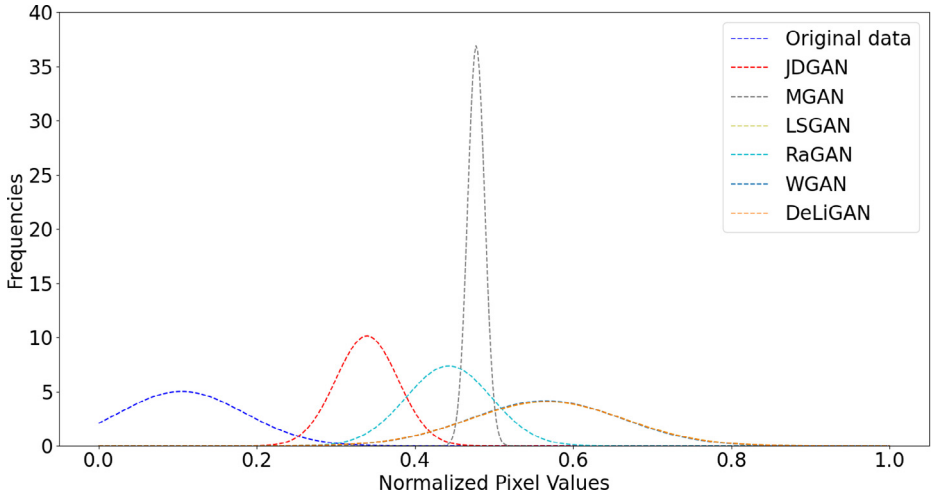


Fig. 7. The overlapping area on the MNIST dataset at the early step. The blue line indicates the original data distribution, and other lines indicate the simulated data distribution, which are generated by LSGAN, WGAN, RaGAN DeLiGAN, MGAN and JDGAN respectively. Only JDGAN holds a non-negligible overlapping area.

more diverse distributions. Similar to the MNIST dataset, the more distributions could also cause generating identical simulated images, for transforming complicated distributions into a relatively simpler distribution could abandon some information of diversity. As for less distributions (e.g., number = 2 in Table 4), such a combination cannot satisfy the rich diversity and the overlapping area could not be enough, thus, the generator tends to generate "realistic" but identical simulation images. In conclusion, we recommend the number of distributions as two different distributions. We now report the *Fréchet Inception Distance* (FID) scores obtained by our JDGAN (Row 1 in Table 5) and baselines (Fig. 10) in Table 6. From Table 6, we can observe that the JDGAN still achieves the smallest score, which further proves the elimination of mode collapse for JDGAN.

Moreover, we compare JDGAN with other GAN-based models on the overlapping area, and the results are shown in Fig. 11. It clear shows that the generating distribution produced by JDGAN holds the largest overlapping area with the original data distribution.

We also test its performance on adequate CIFAR10 training samples (50000 images), and the generated images are shown in Fig. 12. The experimental results further show the effectiveness of JDGAN.

5.4. Other distributions

The Uniform distribution and Gaussian distribution are the most common distributions in statistics, so they are widely used in many GAN variants (e.g, Uniform for the vanilla GAN and DCGAN, Gaussian for the LSGAN and WGAN). To validate the performance of proposed JDGAN on other distributions, we employ the Cauchy distribution, Exponential distribution, Uniform distribution and Gaussian distribution to form the joint distribution, and samples noise code Z from this joint distribution. According

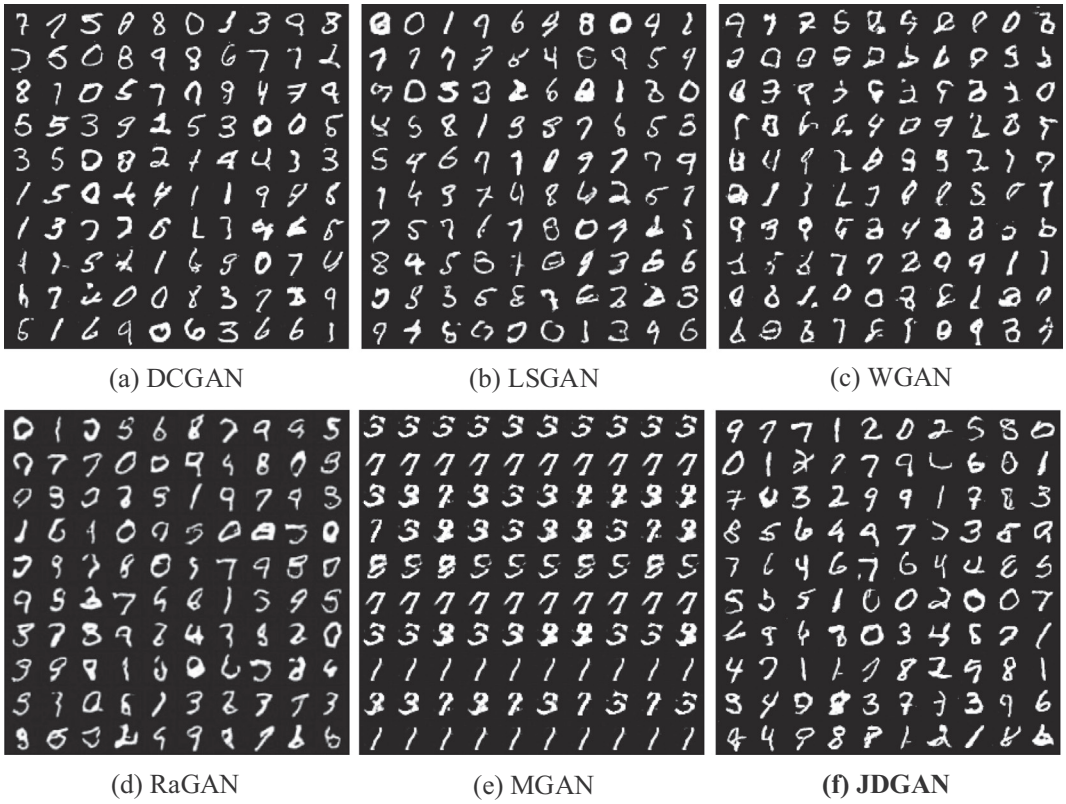


Fig. 8. Generated images produced by baselines and JDGAN on adequate MNIST training samples.

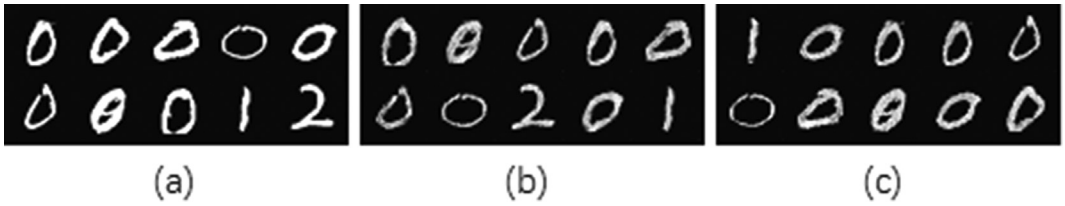


Fig. 9. Sub-figure (a) indicates the training data, and sub-figure (b) indicates the generated data produced by JDGAN. Sub-figure (c) shows the generated data produced by DeLiGAN.

to the previous experimental results, we form the joint distributions of Cauchy and Exponential, Cauchy and Exponential and Uniform, Cauchy and Exponential and Uniform and Gaussian to construct the latent space. In this case, the training data is still the extremely limited samples, and the generated results are shown in Table 7. From Table 7, we can see that the best performance is still in the case of number = 2, larger number resulting in identical instances. We compare the FID score of combination of Cauchy and Exponential (FID_{CE}) with that of combination of Gaussian and Uniform (FID_{GU}), we get FID_{CE} on MNIST and CIFAR10 are 101.4 and 352.1 respectively, which are larger than FID_{GU} 66.2 and 310.8. In this way, the combination of Uniform and Gaussian could be the better choice in our cases.

5.5. Mass spectrometry dataset

Protein is highly complex biochemical entity, and is present in all living organisms. Proteins are the downstream products from genes which carry out almost all essential biological and chemical functions in a living body. Primary structure of a protein is determined by the sequence of specific amino acids (peptide). In recent years, mass spectrometry has become the core analytical technique in studying protein identification and quantification [38]. Mass spectrometry works by ionizing chemical compounds to generate charged molecules or molecule fragments and measuring their mass to charge ratios (m/z). The generated peak spectra reflect the identification and abundance of a specific protein.

In a typical tandem mass spectrometry (MS/MS) experiment, protein mixture is digested by an enzyme into smaller peptides. The sample peptides are then ionized and subjected to MS. As a medical dataset, the main problem is that enough available data collection could be expensive and unrealistic (here we just hold 5 patients and the number of peptide in each patient is 74552). Thus, we hope to use the generative model to generate more simulation data to supplement the original data. We then apply our proposed JDGAN (Eq. (6)) to MS dataset, and the hyper parameters are shown in the right part of Fig. 5. Although there are many features for each patient, we just pick up 6 features (Acq Time, Intensity, Precursor Intensity Acquisition, Apex Time, Elution Peak Width, MS2Counts) because the 6 features correlate to the

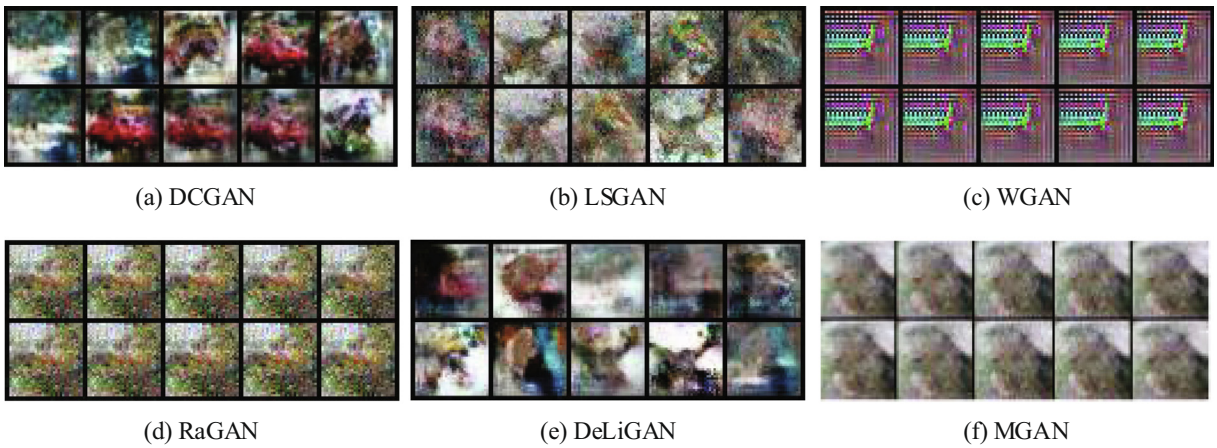
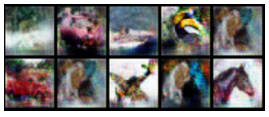
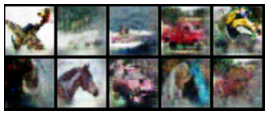
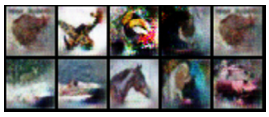
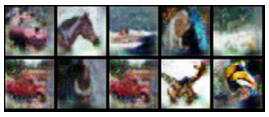


Fig. 10. The generated images are produced by DCGAN, LSGAN, WGAN, RaGAN, DeLiGAN and MGAN, respectively. In this case, the training data only contain 10 samples, and each one represents a specific category. The generator samples noise from standard Gaussian distribution (0, 1) for DCGAN, LSGAN, WGAN, RaGAN and MGAN, and samples noise from a Mixture-of-Gaussians model for DeLiGAN.

Table 4
We jointly employ multiple Gaussian distributions (Simulation Data1) and multiple Uniform distributions (Simulation Data2). The best performance is achieved with the number = 3, which means we employing three same distributions to construct Z.

Number	Simulation Data1	Simulation Data2
2		
3		
4		

differential protein expressions under different biological conditions. In most of the cases, between healthy and disease conditions.

The results are shown in Fig. 13. In Fig. 13, the blue line indicates the threshold, which is a group of mean values calculated by the healthy proteomic values and diseased proteomic values. In most cases, a proteomic value is below the threshold if this proteomic value belongs to the healthy set, while a proteomic value is larger than the threshold as the proteomic value belongs to the diseased set. The red points and green points are simulation data generated by JDGAN, we can see that most simulation data are below the threshold as they belong to healthy set and that are above the threshold as they belong to the diseased set.

Table 5
We jointly employ multiple Uniform and Gaussian distributions, and the best performance is achieved with the number = 1.

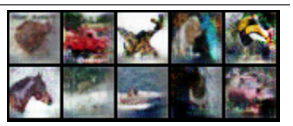
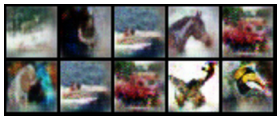
Number	Simulation Data
1	
2	
3	

Table 6
FID scores of generated data shown in Fig. 10.

Models	FID Score
DCGAN	389.6
DeLiGAN	381.9
LSGAN	436.7
WGAN	444.3
RaGAN	453.3
MGAN	456.7
JDGAN	**310.8**

5.6. Discussion

In fact, it is hard to obtain the original data distribution. It becomes even harder for extremely limited data with rich information (e.g., many categories, color or shape information, pixel details). Let $p_z \to z$ and $p_r \to x$ where the dash line indicates model drawing samples from a certain distribution. In the traditional GAN generating mechanism, G always specializes in $z \mapsto x$. Considering $p_z(z)$ is contained in $p_G(z)$, the generated data distribution cannot be complicated to match the original data distribution completely. Under such a scenario, the generated data diversity cannot be guaranteed. JDGAN increases the diversity of generated instances by employing the joint distribution to construct the randomized space. See Figs. 2 and 10.

Note that the crucial point of JDGAN is how to determine the number of distributions. In this paper, we utilize the *Intrinsic Dimension* strategy to evaluate the *Intrinsic Dimensions* of raw data and generated data, because the *Intrinsic Dimension* can loyally reflect the minimal dimension of data manifold [29]. By calculating the *Intrinsic Dimensions* of raw data and generated data (Fig. 2(a)) with Eq. (8), we get $\hat{m}(x)_{raw\ data} = 5.79$ and $\hat{m}(x)_{generated\ data} = 3.30$. This suggests that two distributions may be the optimal choice in

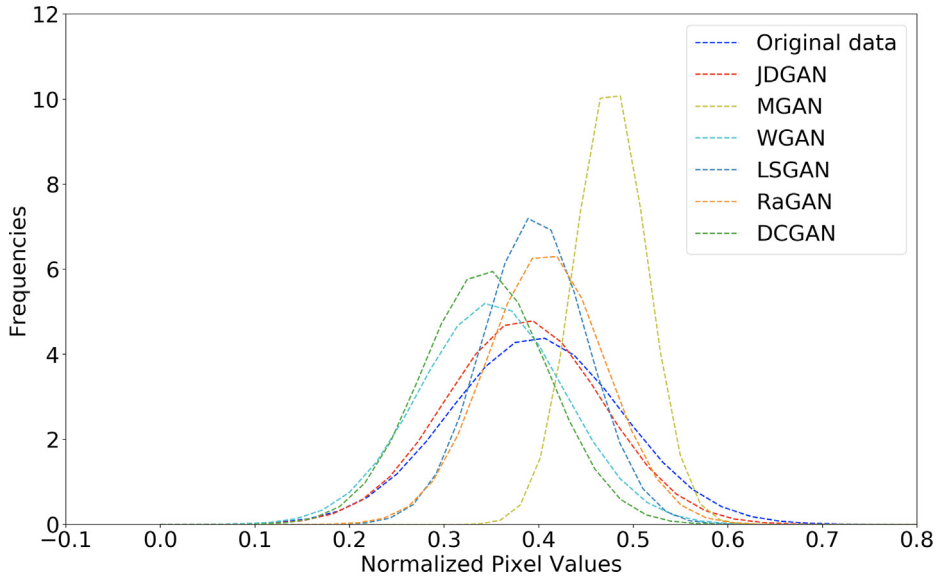


Fig. 11. The overlapping area on CIFAR-10 dataset. The blue line indicates the original data distribution, other lines indicate the simulated data distribution, which are generated by LSGAN, WGAN, RaGAN, MGAN, DCGAN and JDGAN respectively. Only JDGAN holds the largest overlapping area.

our cases, and greater or less than this number would decrease the performance. We still use Eq. (8) to calculate the *Intrinsic Dimensions* of synthetic data produced by JDGAN (Table 5 row1), and get $\widehat{m}(x)_{synthetic\ data} = 5.73$, which is very close to that of raw data. Under such a scenario, the generated distribution is closer to original data distribution (See Figs. 6 and 11), and the diversity of generated data can also be guaranteed (See Tables 2, 3–5 and 7).

6. Conclusion

In this paper, instead of modifying the framework of a GAN model, we explore another direction, which is to modify the randomized space by jointly employing multiple distributions and learn these distributions together with the generator with reparameterization method, to increase the diversity of simulated data

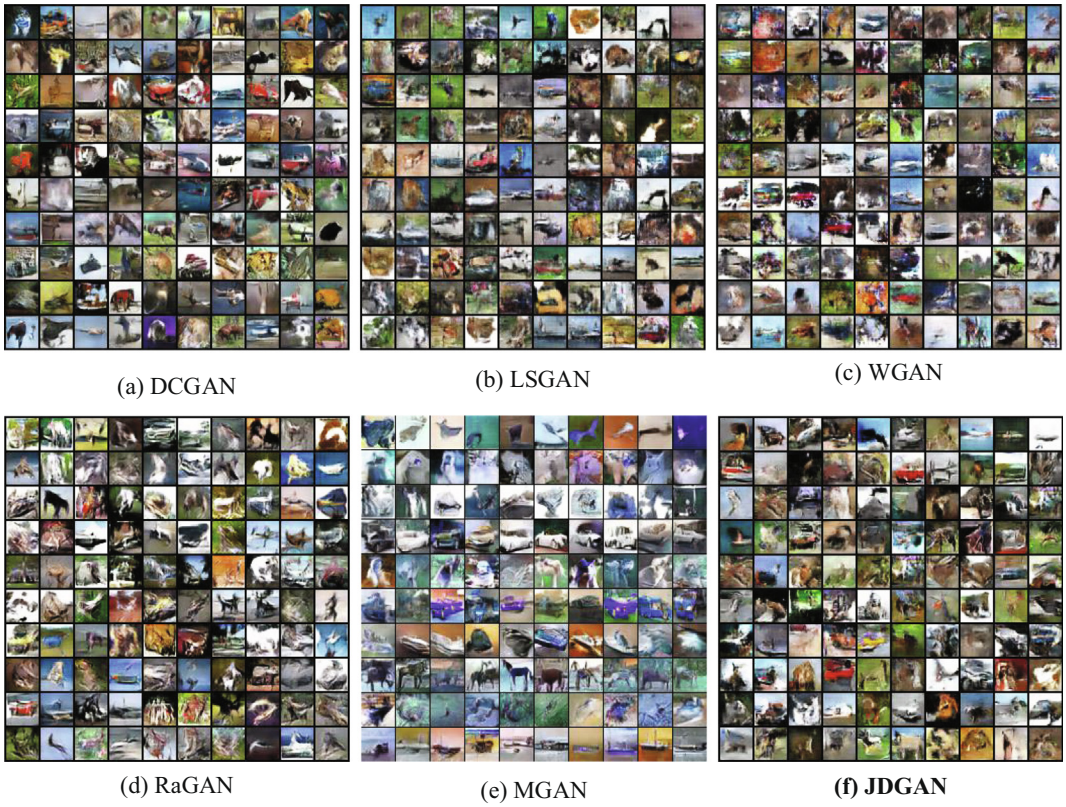
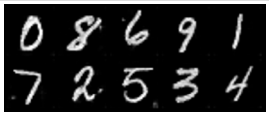
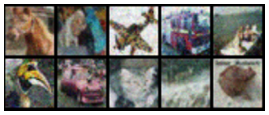
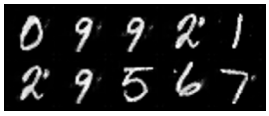
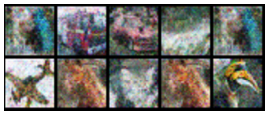
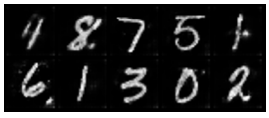
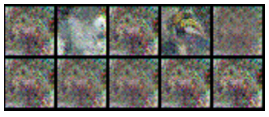


Fig. 12. Generated images produced by GAN variants and JDGAN on adequate CIFAR10 training samples.

Table 7
Number = 2 indicates that one single Cauchy and one single Exponential are employed; number = 3 indicates that one single Cauchy, one single Exponential and one single Uniform are employed; number = 4 indicates that one single Cauchy, one single Exponential, one single Uniform and one single Gaussian are employed.

Number	Simulation Data on MNIST	Simulation Data on CIFAR10
2		
3		
4		

and the overlapping area between the raw data distribution and the generating distribution. Also, we explore how to combine these distributions and display the different performance on different numbers of distributions. Specifically, we jointly employ only multiple Gaussian distributions, only multiple Uniform distributions and the mixture of two distributions respectively, and test each performance. We conducted extensive experiments with MNIST and CIFAR-10 as well as mass spectrometry datasets to validate our proposed JDGAN. The results have shown that our approach is effective and better than other GAN-based models.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported in part by the National Key R$\&$D Program of China (No.2018YFC1604000), in part by the RGC Collaborative Research Fund (CRF) 2018/19 - Group Research Grant (No.:C5026-18G), and in part by the RGC General Research Fund GRF (No.15102015/15M).

References

[1] Luke Metz Alec Radford, Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.
[2] Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stéphane Girard, Michael E. Houle, Ken-ichi Kawarabayashi, Michael Nett, Estimating local intrinsic dimensionality, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 29–38.
[3] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862, 2017.
[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.
[5] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, Yi Zhang, Generalization and equilibrium in generative adversarial nets (gans), in: Proceedings of the 34th International Conference on Machine Learning, vol. 70, 2017, pp. 224–232. JMLR.org.
[6] Léon Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.
[7] Miguel A. Carreira-Perpinan, Mode-finding for mixtures of gaussian distributions, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (11) (2000) 1318–1323.
[8] Constantinos Daskalakis, Paul W. Goldberg, Christos H. Papadimitriou, The Complexity of Computing a Nash Equilibrium, ACM, 2009.
[9] Wei Li, Linchuan Xu, Zhixuan Liang, Senzhang Wang, Jiannong Cao,Chao Ma, and Xiaohui Cui. Sketch-then-edit generative adversarial net-work. Knowledge-Based Systems, page 106102, 2020.
[10] Dominik Maria Endres, Johannes E. Schindelin, A new metric for probability distributions, IEEE Transactions on Information theory (2003).
[11] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, Adam Roberts, Gansynth: Adversarial neural audio synthesis, The International Conference on Learning Representations, 2019.
[12] Jon Gauthier, Conditional generative adversarial nets for convolutional face generation, Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester 2014 (5) (2014) 2.
[13] Arnab Ghosh, Viveka Kulharia, Vinay P. Namboodiri, Philip H.S. Torr, Puneet K. Dokania, Multi-agent diverse generative adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8513–8521.

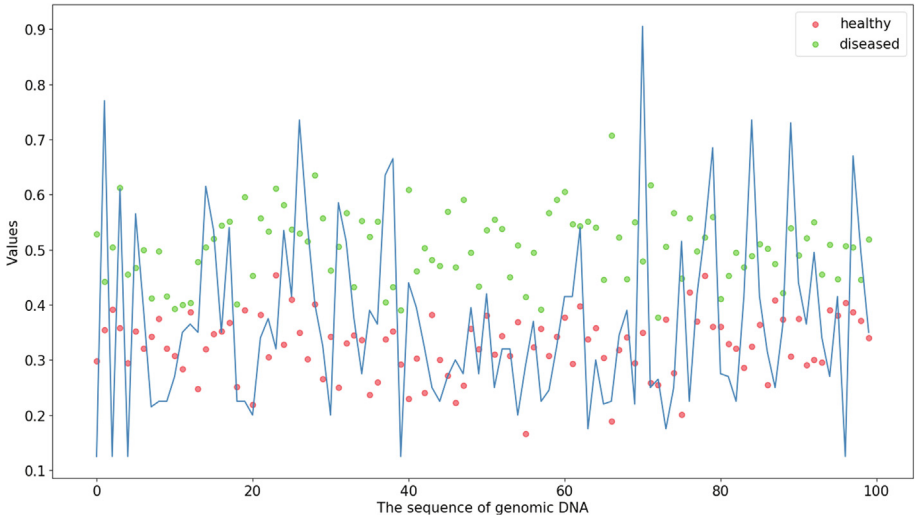


Fig. 13. The green points and red points are belonging to the simulated data generated by JDGAN. The blue line indicates a threshold calculated by original proteomic values included by peptide, which can distinguish whether the peptides within a biological tissue is healthy or not. The tissue is healthy if the corresponding proteomic value is below the threshold, or diseased as the proteomic value is larger than the threshold.

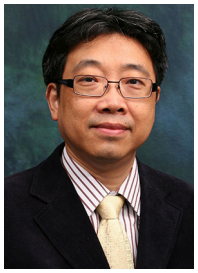
[14] A. Bordes, X. Glort, Y. Bengio, Deep sparse rectifier neural networks, International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron C. Courville, Improved training of wasserstein gans, in: Advances in Neural Information Processing Systems, 2017, pp. 5769–5779.
[17] Swaminathan Gurumurthy, Ravi Kiran Sarvadevabhatla, R. Venkatesh Babu, Deligan: Generative adversarial networks for diverse and limited data, in: CVPR, 2017, pp. 4941–4949.
[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter, Gans trained by a two time-scale update rule converge to a local nash equilibrium, in: Advances in Neural iInformation Processing Systems, 2017, pp. 6626–6637.
[19] C.C. Heyde, Central limit theorem, Encyclopedia of Actuarial Science (2006).
[20] Quan Hoang, Tu Dinh Nguyen, Trung Le, Dinh Phung, Mgan: Training generative adversarial nets with multiple generators, 2018.
[21] Sicong Huang, Qiyang Li, Cem Anil, Xuchan Bao, Sageev Oore, Roger B. Grosse, Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer, The International Conference on Learning Representations, 2019.
[22] Alexia Jolicoeur-Martineau, The relativistic discriminator: a key element missing from standard gan, The International Conference on Learning Representations, 2019.
[23] Mahyar Khayatkhoei, Maneesh K. Singh, Ahmed Elgammal, Disconnected manifold learning for generative adversarial networks, in: Advances in Neural Information Processing Systems, 2018, pp. 7343–7353.
[24] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
[25] Diederik P. Kingma, Max Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114, 2013.
[26] Simon Kohl, David Bonekamp, Heinz-Peter Schlemmer, Kaneschka Yaqubi, Markus Hohenfellner, Boris Hadaschik, Jan-Philipp Radtke, Klaus Maier-Hein (Eds.), Adversarial networks for the detection of aggressive prostate cancer, 2017, arXiv preprint arXiv:1702.08014.
[27] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[28] Dirk P. Kroese, Reuven Y. Rubinstein, Izack Cohen, Sergey Porotsky, Thomas Taimre, Cross-entropy method, in: Encyclopedia of Operations Research and Management Science, Springer, 2013, pp. 326–333.
[29] Elizaveta Levina, Peter J. Bickel, Maximum likelihood estimation of intrinsic dimension, in: Advances in Neural Information Processing Systems, 2005, pp. 777–784.
[30] Mu. Li, Tong Zhang, Yuqiang Chen, Alexander J Smola, Efficient mini-batch training for stochastic optimization, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 661–670.
[31] Yijun Li, Sifei Liu, Jimei Yang, Ming-Hsuan Yang, Generative face completion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, 2017, p. 6.
[32] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, Olivier Bousquet, Are gans created equal? a large-scale study, in: Advances in Neural Information Processing Systems, 2018. pp. 700–709.
[33] Christopher D. Manning, Hinrich Schütze, Foundations of statistical natural language processing, MIT Press, 1999.
[34] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, Stephen Paul Smolley, Least squares generative adversarial networks, in: 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 2813–2821.
[35] Mehdi Mirza, Simon Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784 (2014).
[36] James R. Norris, Markov chains, Number 2. Cambridge University Press, 1998.
[37] Ludger Rüschendorf, The wasserstein distance and approximation theorems, Probability Theory and Related Fields 70 (1) (1985) 117–129.
[38] Eduard Sabidó, Nathalie Selevsek, Ruedi Aebersold, Mass spectrometry-based proteomics for systems biology, Current Opinion in Biotechnology 23 (4) (2012) 591–597.
[39] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, Improved techniques for training gans, in: Advances in Neural Information Processing Systems, 2016, pp. 2234–2242.
[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.
[41] T. Tieleman, G. Hinton, Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. Tech. Rep., Technical report, 2012, p. 31.
[42] Ilya O. Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, Bernhard Schölkopf, Adagan: Boosting generative models, in: Advances in Neural Information Processing Systems, 2017, pp. 5424–5433.
[43] Curtis R. Vogel, Computational methods for inverse problems, volume 23, Siam (2002).
[44] Xiaolong Wang, Abhinav Gupta, Generative image modeling using style and structure adversarial networks, in: European Conference on Computer Vision, Springer, 2016, pp. 318–335.
[45] Xu Bing, Naiyan Wang, Tianqi Chen, Mu Li, Empirical evaluation of rectified activations in convolutional network, Computer Science (2015).
[46] Wei Li, Wei Ding, Rajani Sadasivam, Xiaohui Cui, Chen. PingHis-gan, A histogram-based gan model to improve data generation quality.Neural, Networks 119 (2019) 31–45.
[47] Yu Guoshen, Guillermo Sapiro, Stéphane Mallat, Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity, IEEE Transactions on Image Processing 21 (5) (2012) 2481–2499.
[48] Kai Yu, Tong Zhang, Yihong Gong, Nonlinear learning using local coordinate coding, in: Advances in Neural Information Processing Systems, 2009, pp. 2223–2231.
[49] Ning Zhang, Evan Shelhamer, Yang Gao, Trevor Darrell, Fine-grained pose prediction, normalization, and recognition. arXiv preprint arXiv:1511.07063, 2015.
[50] Li Wei , Fan Li , Wang Zhenyu , Ma Chao , Tacklingmode collapse in multi-generator gans with orthogonal vectors. PatternRecognition 110 107646.

Wei Li received Ph.D. degree in computer science from Wuhan University in 2019. He had visited the University of Massachusetts Boston, and The Hong Kong Polytechnic University as a visiting scholar in 2017 and 2018 respectively. His current research interests include data mining and deep learning with major applications in medical and industrial data.

Linchuan Xu received the B.E. degree in information engineering from Beijing University of Posts and Telecommunications in 2013, and Ph.D. degree from Department of Computing of the Hong Kong Polytechnic University, in 2018. He is currently a post-doctoral researcher of Department of Mathematical Informatics, Graduate School of Information Science and Technology at the University of Tokyo. His current research interests include data mining and deep learning with major applications in network data and medical data.

Zhixuan Liang is a phd candidate of the department of computing, the Hong Kong Polytechnic University. He received his bachelor degree in Shenzhen University and master degree in the Hong Kong Polytechnic University. His research interests are machine learning, multi-agent reinforcement learning.

Senzhang Wang received the B.Sc. and Ph.D. degree in Southeast University, Nanjing, China in 2009 and Beihang University, Beijing, China in 2016 respectively. He is currently an associate professor at College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. His main research focus is on data mining, social computing, and urban computing. He has published more than 70 referred conference and journal papers.



Jiannong Cao received the M.Sc. and Ph.D. degrees in computer science from Washington State University, Pullman, WA, USA, in1986 and 1990, respectively. He is currently the Chair Professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His current research interests include parallel and distributed computing, mobile computing, and big data analytics. Dr. Cao has served as a member of the Editorial Boards of several international journals, a Reviewer for international journals/conference proceedings, and also as an Organizing/Program Committee member for many international conferences.

Dr LAM is currently Associate Professor in the School of Optometry, the Hong Kong Polytechnic University. He completed his BSc training in Optometry with 1st class honors in the HK Polytechnic University. After obtaining his professional license as a Part 1 optometrist, he pursued his Ph.D. training from the same institution in the area of proteomics and myopia. He has more than 10 years' experience in clinical supervision in various optometry clinics locally and overseas. He is among the first to build a tissue-specific retinal proteome using a chick myopia model and propose novel protein regulations in animal myopia models using proteomics coupled to mass spectrometry approach. He also actively serves in the professions as editorial board members of international journals, organizing committee members of international conferences, councilor of The Hong Kong Society of Professional Optometrists (HKSPO), and member of the Optometrists Board, The Government of HKSAR.

Prof. Cui is now a second grade professor of School of Cyber Science and Engineering, Wuhan University. He received his Ph.D. degree in Computer Science and Engineering from Louisville University in 2004. He was Visiting Professor of Louisville University, the head of Food Safety BlockChain Alliance, the vice president of Demonstrational Software School Alliance, the member of High-performance computing special project of National Key R & D Program, the member of advisory committee of Software Engineering Teaching program, the member of CSC, the member of Computing Service of CCF. He was the dean of School of International Software, Wuhan University, and researcher of Computational Data Analysis Department in Oak Ridge National Laboratory.