Pattern Recognition 110 (2021) 107646

Contents lists available at ScienceDirect

Pattern Recognition

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

Tackling mode collapse in multi-generator GANs with orthogonal vectors

Wei Li[a,b,c], Li Fan[d], Zhenyu Wang[d], Chao Ma[d,*], Xiaohui Cui[d,*]

[a] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, Jiangsu, PR China
[b] Jiangsu Key Laboratory of Media Design and Software Technology, Wuxi, Jiangsu, PR China
[c] Science Center for Future Foods, Jiangnan University, Wuxi, Jiangsu, PR China
[d] School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei, PR China

ARTICLE INFO

Article history:
Received 28 February 2020
Revised 8 July 2020
Accepted 6 September 2020
Available online 14 September 2020

Keywords:
GANs
Mode collapse
Multiple generators
Orthogonal vectors
Minimax formula

ABSTRACT

Generative Adversarial Networks (GANs) have been widely used to generate realistic-looking instances. However, training robust GAN is a non-trivial task due to the problem of mode collapse. Although many GAN variants are proposed to overcome this problem, they have limitations. Those existing studies either generate identical instances or result in negative gradients during training. In this paper, we propose a new approach to training GAN to overcome mode collapse by employing a set of generators, an encoder and a discriminator. A new minimax formula is proposed to simultaneously train all components in a similar spirit to vanilla GAN. The orthogonal vector strategy is employed to guide multiple generators to learn different information in a complementary manner. In this way, we term our approach *Multi-Generator Orthogonal GAN* (MGO-GAN). Specifically, the synthetic data produced by those generators are fed into the encoder to obtain feature vectors. The orthogonal value is calculated between any two feature vectors, which loyally reflects the correlation between vectors. Such a correlation indicates how different information has been learnt by generators. The lower the orthogonal value is, the more different information the generators learn. We minimize the orthogonal value along with minimizing the generator loss through back-propagation in the training of GAN. The orthogonal value is integrated with the original generator loss to jointly update the corresponding generator's parameters. We conduct extensive experiments utilizing MNIST, CIFAR10 and CelebA datasets to demonstrate the significant performance improvement of MGO-GAN in terms of generated data quality and diversity at different resolutions.

1. Introduction

Generative Adversarial Networks (GANs) [1] are a class of deep generative models that have been successfully applied to many real-world applications [2–4]. A vanilla GAN consists of two components, a discriminator D and a generator G. G learns to generate new candidates with the same statistical distribution as the training samples, while D estimates the probability that a sample came from the training dataset rather than G. In the training process, both components play a minimax game and contest with each other until they reach the Nash Equilibrium. Although GAN shows powerful generative capabilities [5,6], training robust GAN is still a challenge because it can be easily trapped into the problem of mode collapse in which the generator only concentrates on several, or even only one single, modes rather than all modes.

When we use a neural network to map the real data from original data space into the latent space, the mapped data points usually lie on areas with different sizes. We take the MNIST dataset as the example. Some digits are represented over a very small area and others over a much larger area [7]. It causes that each time the neural network attempts to pick data points from the larger area in the latent space, even though the sampling is random. In other words, such a mapping renders that the model focuses on the data points with large mapping area [8], missing many other modes. Moreover, the Jensen Shannon Divergence is maxed out when the generated data distribution (p_G) and real data distribution (p_r) have a negligible overlapping area, resulting in vanishing gradient. Under such a scenario, the generator apparently fails to capture all the modes of the data, further aggravating mode collapse. The term "mode" in our paper refers to the category and diversity within the category. The category indicates the labels of instances, while the diversity within the category indicates the diverse generated

* Corresponding authors.
E-mail addresses: cs_weili@jiangnan.edu.cn, umass.weili@gmail.com (W. Li), fl2019@whu.edu.cn (L. Fan), zhenyuwang@whu.edu.cn (Z. Wang), whmachao@ieee.org (C. Ma), xcui@whu.edu.cn (X. Cui).

https://doi.org/10.1016/j.patcog.2020.107646
0031-3203/

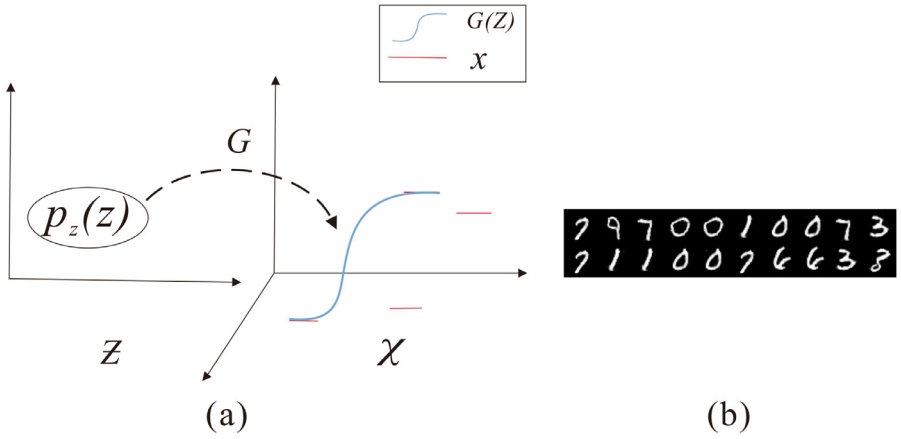


Fig. 1. Illustrative example of transformation from low dimension $\mathcal{Z}$ to high dimension χ. The generator $G(z) : \mathcal{Z} \rightarrow \chi$ with prior $z \sim p_z(z)$ expects to learn all modes of original data, however, such a transformation is not surjective. Sub-figure (a) shows an example of a non-surjective transformation. In sub-figure (a), the disconnected red lines indicate the different original data modes (e.g., categories '0', '1', etc.) while the blue curve indicates the learned modes by G. Sub-figure (b) shows the transformation results. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

instances with the same label. Hence, mode collapse in our paper indicates: 1) the vanilla generator cannot cover all categories; 2) the generated data within the same category tend to be identical. See Fig. 1 for a descriptive illustration of mode collapse. In Fig. 1(b), there exists identical instances and categories '2', '5' and '6' are not captured.

Although many GAN variants have been proposed to tackle this problem, there are still challenges. Wasserstein GAN (WGAN) [9] adopts wasserstein distance to measure the dissimilarity between real data distribution (p_r) and generated data distribution (p_G). However, the weight clipping method adopted by WGAN cannot avoid information loss, resulting in capacity underuse issue [10]. Spectral Normalization GAN (SN-GAN) [11] adopts spectral normalization (λ) to make discriminator Lipschitz continuous, so it can reserve the weight array (W) information to the greatest extent by using $\frac{W}{\sqrt{\lambda}}$. However, since the 1-Lipshitz continuity is achieved by $\frac{W_{ij}}{\sqrt{\lambda_1}}$, the generated data would be inevitably mapped into a specific color sub-space, rendering all generated objects taking the same color. This can also be viewed as mode collapse.

The very recent attempt is to employ multiple generators to address mode collapse, motivated by the limitations of single generator for learning different modes [12,13]. Multiple Generators GAN (MGAN) [13] employs a mixture of generators and adopts "shared parameters" method to simultaneously train all generators. Since there are no restrictions enforcing generators to learn all modes, these generators tend to learn the same modes, resulting in producing identical instances. Considering the significance of these challenges and the potential benefits of overcoming them, it is worth to develop a novel approach to addressing the problem of mode collapse.

In this study, we propose a novel approach to training multi-generator model with orthogonal vectors. The distributions of generated data from multiple generators are jointly induced to a mixture distribution for matching the original data distribution while encouraging each generator to learn different information of original data. To achieve this, we employ orthogonality strategy [14,15] to construct orthogonal vectors based on the outputs of generators, and enforcing orthogonal vectors contain different information during training. Hence, we term our approach *Multi-Generator Orthogonal GAN* (MGO-GAN). Initially, the generated data of all generators are fed into an extra encoder to obtain feature vectors. Then, we calculate the inner product between each pair of feature vectors to obtain the orthogonal value, which loyally reflects the correlation between two vectors. The lower the orthogonal value is, the more information the two vectors hold. The lower orthogonal value also demonstrates how different information has been learnt by the two generators. Therefore, we need to minimize the orthogonal value during training, and this is achieved by back-propagation along with minimizing the generator loss. After that, the orthogonal value is integrated with the generator loss to jointly update the corresponding generator's parameters, rendering this generator learning such information while other generators hold less. A novel loss function is able to be established among a set of generators, an encoder and a discriminator. Furthermore, we provide theoretical analysis that the Jensen-Shannon (JS) divergence [16] between the mixture distribution and the original data distribution is minimal, and the orthogonal value between any two feature vectors is also minimal.

In summary, the major contributions of this study are described as follows:

- This paper proposes a novel MGO-GAN model which learns a mapping function parameterized by multiple generators from the randomized space to the original data space, overcoming the problem of mode collapse.
- This paper utilizes the back-propagation to minimize the orthogonal value in GAN and combine the orthogonal value with the generator loss to jointly update the parameters of generator from both theoretical and empirical perspectives, offering new insights into the success of MGO-GAN.
- Through comprehensive experiments on three datasets with different resolutions, we demonstrate the effectiveness of the proposed approach.

The rest part of this paper is organized as follows. In Section 2 existing works are discussed. We review the GAN model and orthogonal vectors in Section 3. MGO-GAN is demonstrated in detail in Section 4. In Section 5 we show our experimental results. Section 6 serves as our conclusion.

2. Related work

The problem of mode collapse has attracted a lot of attentions, and many GAN variants are proposed to tackle this problem. According to the strategies these algorithms adopt, here we group these studies into two categories.

Loss Function Creativity. The generated data distribution p_G cannot completely match the original data distribution p_r, which is caused by the joint consequence of JS divergence and the negli-

gible overlapping area between p_G and p_r. If this overlapping area is negligible, JS divergence becomes a constant (i.e., log2). The constant loss cannot deduce a meaningful derivation for updating the generator's parameters, such that G cannot guarantee producing plausible data. In this way, researchers attempt to modify the loss function to address this problem. Wasserstein GAN [9] replaces the JS divergence with Wasserstein distance, because Wasserstein distance [17] can measure the dissimilarity between p_G and p_r even though the supports of them are disjoint. For approaching the Wasserstein distance, WGAN adopts Kantorovich-Rubinstein duality to transform the standard Wasserstein distance into a discriminator function (i.e., $f_w^*(x)$), and the 1-Lipshitz continuity can guarantee this transformation. For satisfying 1-Lipshitz continuity, $\frac{\partial f_w^*(x)}{\partial x}$ has been limited to a specific range (e.g., $(-0.01, 0.01)$). However, the optimal strategy is either to take the largest value (i.e., 0.01) or take the smallest value (i.e., -0.01) for all parameters under such a scenario, causing gradients vanishing or explosion. Moreover, only certain optimizers (e.g., *RMSProp* or *SGD*) in WGAN are suitable for optimization [9] but momentum based ones (e.g., *Adam*) may turn the gradients negative.

Takeru et al. [11] proposed a novel weight normalization method to achieve the 1-Lipshitz continuity, and it is termed as spectral normalization GAN (SN-GAN). SN-GAN uses this spectral normalization to control the Lipschitz constant of the discriminator function f. To calculate the spectral normalization, SN-GAN adopts Singular Value Decomposition (SVD) to get the largest singular value and views this value as the spectral normalization, and uses $\frac{W_{ij}}{\sqrt{\lambda_1}}$ to enforce the discriminator 1-Lipshitz continuity in which λ_1 indicates the largest singular value of W (weight matrix of discriminator) and W_{ij} indicates a specific element within W. However, such a strategy does not completely correspond to penalising the spectral norm [18]. In addition, SN-GAN requires that each layer of the network satisfies 1-Lipshitz continuity handled by $\frac{W_{ij}}{\sqrt{\lambda_1}}$, such a constraint causes that the generated instances are mapped into a specific color sub-space, rendering all generated objects taking the same color. This can also be viewed as mode collapse.

To produce diverse generated data, Localized GAN (LGAN) [19] employs local coordinate chart $G(x, z)$ around data point x to learn local geometry near x, with its local coordinates z drawn from a random distribution $p_z(z)$. The tangent vectors located at x are employed to guarantee learning success. Note that a linear tangent space formed by N tangent vectors would collapse if it is dimensionally deficient. To avoid this, the orthonormal counterpart is utilized to prevent the collapse of the tangent space. In fact, LGAN requires many assumptions which are not practical in real-world applications. For example, the origin of the local coordinates z is assumed locating at the given point x. It is not easy to achieve this in practice, given that $p_z(z)$ is usually a simple, easy-to-sample distribution while $p_r(x)$ is a complex, high-dimensional distribution. The noise code cannot fill in the whole high-dimensional distribution, resulting in no noise code around many points. In addition, the generator of LGAN is made up of encoder-decoder, the generated data quality cannot be guaranteed (generated images are blurred) [7].

Architecture Creativity. Given the fact that the single generator cannot capture all modes during training, some researchers attempt to employ multiple generators to learn different information. Tolstikhin et al. [20] trained a mixture of generators with boosting techniques named AdaGAN. At every step, AdaGAN adds a new component into a mixture model by running a GAN algorithm on a reweighted sample, and it aggregates many potentially weak individual predictors to form a strong composite predictor. However, sequentially training multiple generators needs more extra cost. In addition, AdaGAN assumes that a single generator can produce impressive images of some modes, so a mixture of generators can cover the whole data space. Such an assumption is impractical in the real world [13]. Arora et al. [21], 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. [12] proposed MAD-GAN, which is a multi-agent GAN architecture incorporating multiple generators and one multi-class discriminator. Not only can this discriminator detect whether a sample is fake or not, but also can predict which generator produces the sample. The loss function in this study, however, focuses on detecting whether a sample is fake or not and does not directly encourage generators to produce diverse instances.

MGAN is recently proposed by Hoang et al. [13], which employs a set of generators $G_{1:K}$, a single discriminator D and an extra classifier C to construct the architecture of MGAN. To train MGAN, a novel minimax formulation is developed to establish among those components. In MGAN, K generators altogether induce a mixture distribution named p_{model}. An index μ indicates μth generator $G_\mu(z)$, and μ follows a multinomial distribution. MGAN utilizes JS divergence to handle equilibrium and mode collapse problems during training. For K generators, MGAN aims to maximize the JS divergence among these generators. Generators implicitly hold different information if JS divergence is log2 among $p_{G_1}, p_{G_2}, \ldots, p_{G_K}$. Decreased JS divergence between p_r and p_{model} indicates generated data approaching to original data. Since MGAN adopts "shared parameters" strategy to simultaneously train all generators, it tends to hold the same parameters. Under such a scenario, all generators in MGAN produce identical instances. In other words, MGAN cannot well address the problem of mode collapse. More details are shown in Appendix A.

3. Preliminaries

3.1. Generative adversarial network

Although Generative Adversarial Networks (GANs) [1] were introduced in the introduction, we formally describe it below to establish continuity. GANs were developed by Goodfellow as a novel generative model to simultaneously train a generator G and a discriminator D using the following function:

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

where D indicates the discriminator, G indicates the generator, and both are neural networks. x comes from a distribution $p_r(x)$ underlying the original 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 with $(-1, 1)$ or Gaussian distribution with $(0, 1)$. G tries to fool D into accepting its outputs as real by maximizing its score $D(G(z))$, and this is achieved by the following optimization function.

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

Moreover, the discriminator D takes an input from either the original dataset or the generator and produces a probability that the input comes from the original dataset rather than G. In general, the discriminator D strives to minimize the score it assigns to the generated data $G(z)$ by minimizing $D(G(z))$ and maximize the score it assigns to the original data x by maximizing $D(x)$. In this way, the optimization function for D is shown as follows.

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

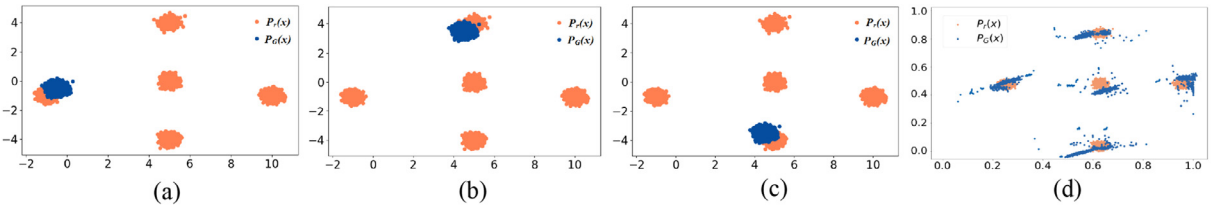


Fig. 2. $P_r(x)$ is the 5-Gaussian (coral clusters) and we assume in this case each Gaussian indicating a single mode. The example shows that the vanilla GAN maps the random Gaussian to a certain Gaussian at different epochs (steel-blue cluster in sub-figures (a), (b) and (c)). Our proposed MGO-GAN captures all the five Gaussian modes as shown in sub-figure (d). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In this work, the discriminator and the generator are alternatively optimized, and JS divergence is utilized to measure the difference between p_r and p_G. JS divergence reaches its lowest value as the discriminator and the generator reach the Nash Equilibrium where $D(G(z)) = D(x) = 0.5$. The GAN model gets converged under such a scenario.

3.2. Mode collapse

When a network maps a dataset into a latent space, the mapped areas are different [7]. To obtain the generalization capability of GAN, we usually adopt the stochastic method (i.e., stochastic gradient descent optimization or stochastic sampling method) as training strategy. However, such a strategy implicitly indicates that each time the neural network more easily tends to pick data points from the larger areas than to pick data points from the smaller areas in the latent space, given that the mapped data points lie on areas with different sizes. Considering an extreme scenario, the GAN model each time picks data points from the largest region. Under such a scenario, the remaining modes lying on other areas are missed even though G produces realistic-like data. Moreover, when G maps noise sampled from randomized space $\mathcal{Z}$ into the original data space, such a mapping is not surjective [8,22]. This further deteriorates the capability of G to capture different modes.

Moreover, the discriminator D actually discourages the GAN model to capture all modes. On the one hand, D cannot give how many modes are captured in real-world applications, because D is only used to evaluate the probability that a sample is from the original dataset rather than G. Therefore, judging whether all modes are captured by G becomes a non-trivial task. On the other hand, D always maximizes the probability of assigning the correct label to both the original data and generated data (See Eq. (3)) during training [1]. In other words, D judges all samples generated by G as fake (low probability for $D(G(z))$) even though G captured some original data modes. To fool D, G searches for other modes and tries to learn them. Since the mapping is not surjective, GAN repeats such a cat-and-mouse game and cannot stop the training process. Note that there is no countermeasure in Eq. (1) that explicitly forces generator G to get out from this scenario. Hence, G has the tendency to produce identical but safe instances rather than diverse but unsafe samples. Fig. 2 shows a cat-and-mouse example. Moreover, we also validate our proposed MGO-GAN (5 generators are employed) in the same case. Obviously, our proposed MGO-GAN captures all the five Gaussian modes (See sub-figure (d)).

3.3. Orthogonal vectors

Orthogonal vectors [14,15] are referred to two or more vectors whose inner product is 0, i.e., the vectors are perpendicular to each other. If two non-zero vectors are orthogonal, they must be linearly independent. The definition of orthogonal vectors is shown as follows.

Assuming $\mathcal{V}$ is a finite dimensional linear space in $\mathbb{R}$, and vectors α and β ($\alpha, \beta \in \mathbb{R}$ and $\alpha \neq \beta$) are two non-zero vectors which are defined in $\mathcal{V}$. In this way, $\mathcal{V}$ is defined as the Euclidean space, and the *cosine* of intersection angle ($\mathcal{O}(\alpha, \beta)$) between α and β is defined as the orthogonal value which can be defined as Eq. (4).

$$\mathcal{O}(\alpha, \beta) = \left| \frac{(\alpha, \beta)}{|\alpha||\beta|} \right| \tag{4}$$

$\mathcal{O}(\alpha, \beta)$ can loyally reflect the correlation between α and β. The smaller the value of $\mathcal{O}(\alpha, \beta)$ is, the more different information the two vectors contain. The motivations of adopting orthogonal vectors are twofold:

1. **The orthogonal value indicates how different information has been learnt by generators.** The two vectors (α and β) are produced by an encoder which takes the outputs of two different generators (G_i and G_j) as its input. The smaller the value of $\mathcal{O}(\alpha, \beta)$ is, the more information the two generators have learned. Therefore, $\mathcal{O}(\alpha, \beta)$ could be used as an indicator to measure the difference of information learned by the two generators. We take the MNIST dataset as the example. Assuming there are 2 generators (G_1.G_2), G_1 has learned the digits '0'-'4'. If G_2 has learned the same digits, the two encoded vectors hold the homogeneous feature information after we feed all the generated instances into the encoder. Under such a scenario, $\alpha \approx \beta$ and $\mathcal{O}(\alpha, \beta)$ holds a largest value. If G_2 has learned different digits (e.g., figures '5'-'9'), the encoded vector (α) hold totally different feature information, i.e., $\alpha \neq \beta$. Under such a case, $\mathcal{O}(\alpha, \beta)$ holds a small enough value. In this paper, we assume this small enough value as δ. In this way, the orthogonal value loyally reflects how different information has been learnt by generators.
2. **The orthogonal value can be integrated into the training of GAN.** To guarantee that the orthogonal value can be minimized in the training of GAN, we minimize the orthogonal value ($\mathcal{O}(\alpha, \beta)$) along with minimizing the generator loss with back-propagation, and integrate the orthogonal value ($\mathcal{O}(\alpha, \beta)$) with JS divergence ≥ 0 to jointly update the parameters of generators. This is because the orthogonal value ($\mathcal{O}(\alpha, \beta)$) satisfies nonnegativity and Cauchy inequality [23] in the Euclidean space.

In this manner, MGO-GAN enables multiple generators to learn different information in a complementary and efficient way. In the next section, we present our proposed MGO-GAN and give a theoretical analysis, essentially showing that the training criteria allows multiple generators to learn different information.

4. MGO-GAN

As described in previous analysis, employing one single generator is hard to learn all modes of original data. To address this problem, MGO-GAN employs multiple generators and orthogonal vectors to learn those modes. In this way, there are two important

issues to be addressed: 1) How the generators learn different information; 2) Will MGO-GAN converges to an equilibrium.

4.1. Theoretical analysis

We employ a set of generators $G_{1:K}$ to jointly build a mapping function that can map the random noise code z from randomized space $\mathcal{Z}$ into the data space χ. We define a prior distribution on $\mathcal{Z}$ as $p_z(z)$ (e.g., Uniform with $(-1, 1)$ or Gaussian with $(0, 1)$) where the generators sample noise from. The original data follows a specific distribution which we term $p_r(x)$. As a GAN variant, MGO-GAN still plays a minimax game, i.e., maximizing the discriminator D and minimizing the generators $G_{1:K}$. Different from the vanilla GAN, each generator in MGO-GAN is encouraged to focus on such the information while other generators do not hold. In this way, the objective function of MGO-GAN can be defined as follows:

$$\begin{aligned}\min_{G_{1:K}}\max_{D} V(G_{1:K}, D) &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] \\ &+ \frac{1}{K}\sum_{i=1}^{K}\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G_i(z)))] \\ &+ \frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)), E(G_j(z))) \end{aligned} \tag{5}$$

where K indicates the number of generators and λ indicates a coefficient. E indicates an Encoder which is used to extract the feature vectors of generated data produced by K generators, such that the feature vectors are mapped into the same space to be calculated the corresponding orthogonal value. For the last term $\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)), E(G_j(z)))$, i, j belong to the range of [1, K] while $i \neq j$. In MGO-GAN, D still tries to maximize the probability of assigning the correct label to both training and generated samples, and G tries to fool D into accepting its outputs as real data by maximizing its score $D(G_{1:K}(z))$. The optimization formulas for D and $G_{1:K}$ are shown as follows:

$$\begin{aligned}\max_{D}\ V(G_{1:K}, D) &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] \\ &+ \frac{1}{K}\sum_{i=1}^{K}\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G_i(z)))]\end{aligned}$$

$$\begin{aligned}\min_{G_{1:K}}\ V(G_{1:K}, D) &= \frac{1}{K}\sum_{i=1}^{K}\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G_i(z)))] \\ &+ \frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)), E(G_j(z)))\end{aligned}$$

Each generator G_i implicitly defines a probability distribution p_{G_i} as the distribution of the generated samples $G_i(z)$, K generators altogether induce a mixture distribution to fit p_r. We consider the optimal discriminator D for generators $G_{1:K}$. Assuming the discriminator D has enough capacity, we show below that the optimal discriminator D is at the equilibrium point $p_r = \frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}$.

Proposition 1. *For $G_{1:K}$ fixed, the optimal discriminator D is:*

$$D^*(x) = \frac{p_r}{p_r + \frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}} \tag{6}$$

Proof. The training criterion for the discriminator D, given a mixture of generators G_1, G_2, ..., G_K, is to maximize the quantity $V(G_{1:K}, D)$.

$$\begin{aligned}V(G_{1:K}, D) &= \int_x p_r(x)\log D(x)dx \\ &+ \frac{1}{K}\Big(\int_z p_z(z)\log[1-D(G_1(z))]dz \\ &+ \int_z p_z(z)\log[1-D(G_2(z))]dz \\ &+ \ldots + \int_z p_z(z)\log[1-D(G_K(z))]dz\Big) \\ &= \int_x p_r(x)\log D(x) + \frac{1}{K}p_{G_1}(x)\log[1-D(x)] \\ &+ \frac{1}{K}p_{G_2}(x)\log[1-D(x)] \\ &+ \ldots + \frac{1}{K}p_{G_K}(x)\log[1-D(x)]dx \\ &= \int_x p_r(x)\log D(x) \\ &+ \frac{p_{G_1(x)}+\ldots+p_{G_K(x)}}{K}\log[1-D(x)]dx\end{aligned}$$

□

We compute the partial derivation of $\frac{\partial V(D(x))}{\partial D(x)}$, and get $\frac{p_r}{D(x)} = \frac{1}{K}\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{1-D(x)}$. In this way, it can be easily seen that D achieves its maximum in [0,1] at $\frac{p_r}{p_r+\frac{p_{G_1(x)}+p_{G_2(x)}+\ldots+p_{G_K(x)}}{K}}$. Hence, Eq. (6) is obtained, and the optimum of Eq. (6) is $\frac{1}{2}$ when $p_r = \frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}$. Based on Proposition 1, we substitute Eq. (6) into the Eq. (5), we get:

$$\begin{aligned}C(G_{1:K}) &= \max_D V(G_{1:K}, D) \\ &= \mathbb{E}_{x\sim p_r(x)}[\log D^*(x)] \\ &+ \frac{1}{K}\sum_{i=1}^{K}\mathbb{E}_{z\sim p_z(z)}[\log(1-D^*(G_i(z)))] \\ &+ \frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)), E(G_j(z))) \\ &\approx -2\log 2 + KL\left(p_r||\frac{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{2}\right) \\ &+ KL\left(\frac{p_{G_1}+\ldots+p_{G_K}}{K}||\frac{p_r+\frac{p_{G_1}+\ldots+p_{G_K}}{K}}{2}\right)+\delta \\ &= -2\log 2 + 2JSD\left(p_r||\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}\right)+\delta\end{aligned} \tag{7}$$

where KL indicates Kullback-Leibler divergence [24] and JSD indicates the Jensen-Shannon divergence [25]. From Eq. (7), we can see that the training criterion is determined by both components (i.e., generators ($G_{1:K}$) and orthogonal value ($\mathcal{O}(E(G_i(z)), E(G_j(z)))$)). In the early steps of training, the difference between p_r and $\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}$ is large. $\mathcal{O}(E(G_i(z)), E(G_j(z)))$ is also large. This is because the generated data are basically noise in the first few epochs. As the epochs increase, generated data approximates original data and the orthogonal value becomes small. This is achieved by training criterion with back-propagation through $\nabla_\theta\left(\frac{1}{K}\log(1-D(G_{i_\theta}(z))) + \mathcal{O}(E(G_{i_\theta}(z)), E(G_{j_\theta}(z)))\right)$ in a similar way as vanilla GAN, $j \in 1{:}\ K$ and $j \neq i$. The smaller orthogonal value reflects generators learning different information. If each generator learns the information while other generators do not hold, the orthogonal value is the minimum. Here, we assume the minimum as δ. A detailed derivation of Eq. (7) is shown in Appendix B.

Theorem 1. *The global minimum of the virtual training criterion $V(G)$ is achieved if and only if $p_r = \frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}$ and each generator*

captures information which other generators do not capture. At that point, V(G) achieves the optimal value $-2\log 2+\delta$.

Proof. For $p_r=\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}$, $D^*(x)=\frac{1}{2}$ according to Eq. (6). Then, $p_r=\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}$ is substituted into the Eq. (7), we get $C(G)=-2\log 2+JSD(p_r||\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K})+\mathcal{O}(E(G_i(z)),E(G_j(z))), i,j\in[1,K], i\neq j$. The JS divergence between original data distribution and mixture distribution of generated data is always non-negative and reaches the minimum 0 only when $p_r=\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}$, $JSD(p_r||\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K})=0$. In MGO-GAN, the orthogonal value is minimized along with minimizing the generator loss with $\nabla_\theta\left(\frac{1}{K}\log(1-D(G_{i_\theta}(z)))+\mathcal{O}(E(G_{i_\theta}(z)),E(G_{j_\theta}(z)))\right)$. In other words, the orthogonal value is gradually decreased when the generator loss is decreased. Note that the orthogonal value is then integrated with generator loss to jointly update the parameters of this generator. The generator gradually holds such an information while other generators do not hold. The orthogonal value would be minimum when JS divergence reaches the minimal status. Under such a scenario, $V(G)=-2\log 2+\delta$. □

Note that one single generator usually causes that the generated data distribution p_G holds a negligible (or even none) overlapping area with original data distribution p_r. Given $V_{vanilla}(G)=-2\log 2+2JSD(p_r||p_G)$, $JSD(p_r|p_G)=\log 2$ if p_G and p_r holds a negligible overlapping area [1,26]. Under such a scenario, $V_{vanilla}(G)$ equals to 0. In other words, the gradients of generator G disappear, and G cannot get updated. However, our proposed MGO-GAN can avoid this problem. We now state this properly in the following Theorem.

Theorem 2. *The gradients vanishing problem could be avoided in MGO-GAN, and the generators still get updated even though the mixture distribution (i.e.,* p_m*) deduced by all generators has a negligible overlapping area with original data distribution* p_r.

Proof. For MGO-GAN, $V(G)=-2\log 2+2JSD(p_r||p_m)+\mathcal{O}(E(G_i(z)),E(G_j(z)))$. Assuming the mixture distribution p_m has a negligible overlapping area with original data distribution p_r at a certain epoch, $JSD(p_r||p_m)$ still equals to $\log 2$, such that $-2\log 2+2JSD(p_r||p_m)$ equals to 0. However, $\mathcal{O}(E(G_i(z)),E(G_j(z)))$ is not equivalent to 0 under such a scenario. Hence, $V(G)$ is also not equivalent to 0, such that the generators still get updated. Therefore, the gradients vanishing problem could be avoided in our proposed MGO-GAN. □

The pseudo-code of MGO-GAN algorithm is formally presented in Algorithm 1. Our generators and discriminator architectures are from vanilla DCGAN [27], and the details of components still adopt Conv-BatchNorm-ReLu [28] (Generators $G_{1:K}$) and Conv-BatchNorm-LeakyReLu [29] (Discriminator D). In Algorithm 1, E indicates an *Encoder* which abstracts the feature information of generated instances. $\mathcal{O}_{idx}$ indicates the orthogonal value among the current generator (G_{idx}) and other generators (G_i, $i\neq idx$ and $i\in[1,K]$), i.e., $\mathcal{O}(D(G_{idx}(z)),D(G_{1:idx-1,idx+1:K}(z)))$. After that, we integrate the orthogonal value $\mathcal{O}_{idx}$ with the original generator loss to update the parameters of the corresponding generator (G_{idx}). The other generators also adopt the same strategy to calculate the orthogonal value and to be updated. The architecture of our proposed MGO-GAN is shown in Fig. 3.

4.2. Convergence of Algorithm 1

Proposition 2. *Assuming all generators deduce a mixture of distribution* $p_m=\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}$ *and D is in the optimal status given* $G_{1:K}$ *if D has enough capacity, then* p_m *converges to* p_r.

Algorithm 1 MGO-GAN.

Input: Original samples; noise z;
Output: Simulation data.

def Cal-Orthogonal(idx,K):
  $\mathcal{O}=0$
  for 1:K **do**
    $\mathcal{O}+=\left|\frac{(E(G_i),E(G_{idx}))}{|E(G_i)||E(G_{idx})|}\right|$, idx ≠ i
  return $\mathcal{O}$

Adam optimizer and *BCE* loss function are adopted. The quantity of generator is K, and m indicates mini-batch samples.
E represents the Encoder.
for number of iterations **do**

- Sampling minibatch of m noise samples $z^1,\ldots,z^m$ from $p_z(z)$.
- Sampling minibatch of m original samples $x^1,\ldots x^m$ from real dataset.
- Updating the parameters of D by ascending its stochastic gradient.
- $\nabla_{\theta_D}\frac{1}{m}\sum_1^m\{\log D(x^{(i)})+\frac{1}{K}\log(1-D(G_{1:K}(z^{(i)})))\}$.
- Sampling minibatch of m noise samples $z^1,\ldots,z^m$.
- Iteratively calculating the orthogonal value $\mathcal{O}_{idx}=$ Cal-Orthogonal(idx,K) on each generator (G_{idx}), $idx\in[1:K]$.
- Updating the generators' parameters and minimizing the orthogonal value by descending its stochastic gradient.
- $\nabla_{\theta_{G_{idx}}}\frac{1}{m}\sum_1^m\{\frac{1}{K}\log(1-D(G_{idx_\theta}(z^{(i)}))+\mathcal{O}(E(G_{idx_\theta}(z^{(i)})),E(G_{j_\theta}(z^{(i)})))\}$, $j\in 1:K$ and $j\neq idx$.

end for

Proof. Considering $V(G_{1:K},D)=U(p_m,D)$ is a convex function in p_m obtained from the optimization functions, i.e., Eqs. (6) and (7). The subderivatives of a supremum of convex functions include the derivative of the function at the point in which the maximum is obtained. That is to say, if $f(x)=sup_{\alpha\in\mathcal{A}}f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x)\in\partial f$ if $\beta=argsup_{\alpha\in\mathcal{A}}f_\alpha(x)$. According to Sion minimax theorem [30], let p_m and p_r be non-void convex and compact subsets of two linear topological spaces, then $\min_{G_{1:K}}\max_D V(G_{1:K},D)\geq\max_D\min_{G_{1:K}}V(G_{1:K},D)$. We can deduce each time computing a gradient descent update for p_m given a set of generators $G_{1:K}$ and an optimal D. $sup_D U(p_m, D)$ is convex in p_m with a unique global optimum as proven in optimization process, therefore with sufficiently small updates of p_m, p_m converges to p_r. □

5. Experiments

For the experiments, three commonly used public datasets, MNIST, CIFAR10 and CelebA, are studied. The MNIST dataset contains 60,000 gray-scale images with size 28 × 28. CIFAR10 and CelebA respectively contain 50,000 and 10,000 RGB images. In CIFAR10, the size of each image is 3 × 32 × 32, and that size is 3 × 178 × 218 in CelebA. Both MNIST and CIFAR-10 datasets are labeled with 10 categories. The CelebA dataset has no explicit labels.

Based on the selected datasets, the neural network settings of MGO-GAN components are shown in Fig. 4. Note that each generator in MGO-GAN holds the same setting. They sample noise from the standard Gaussian distribution (0, 1) ($p_z(z)$). Since **WGAN** [9], **SN-GAN** [11], **LGAN** [19] and **MGAN** [13] are representatives of GAN variants on addressing the problem of mode collapse, they are employed as the baselines in this study. For all models, we adopt the same hyperparameters (e.g., $Epoch=200$) and the same running environment (e.g., *Pytorch* framework). As to the length of encoded

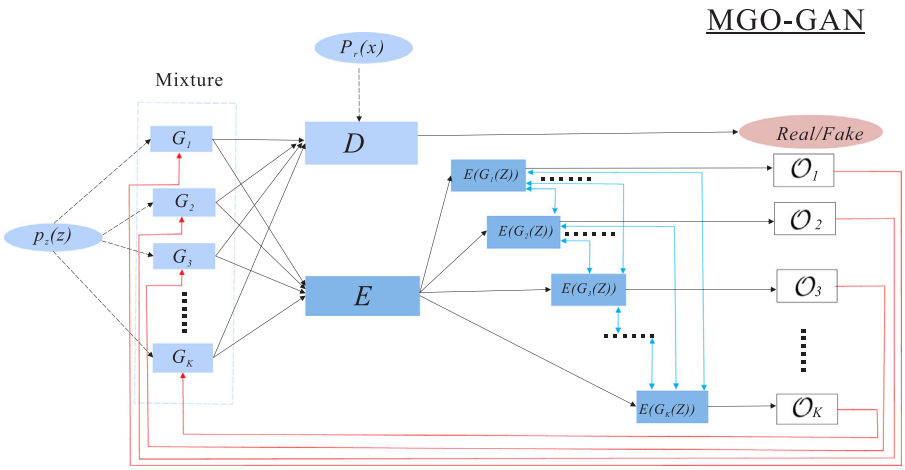


Fig. 3. The architecture of our proposed MGO-GAN in which a set of generators ($G_{1:K}$), an Encoder (E) and a discriminator (D) are employed. Dotted arrows indicate sampling from a specific distribution. E can help produce feature vectors ($E(G_i(z))$), which is used to calculate the orthogonal values ($\mathcal{O}_i$) via inner product that is marked by the cyan line. The orthogonal value is integrated into the corresponding generator loss to update the corresponding generator's parameters, which is marked by the red line. All generators deduce a mixture distribution to approximate the original data distribution. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

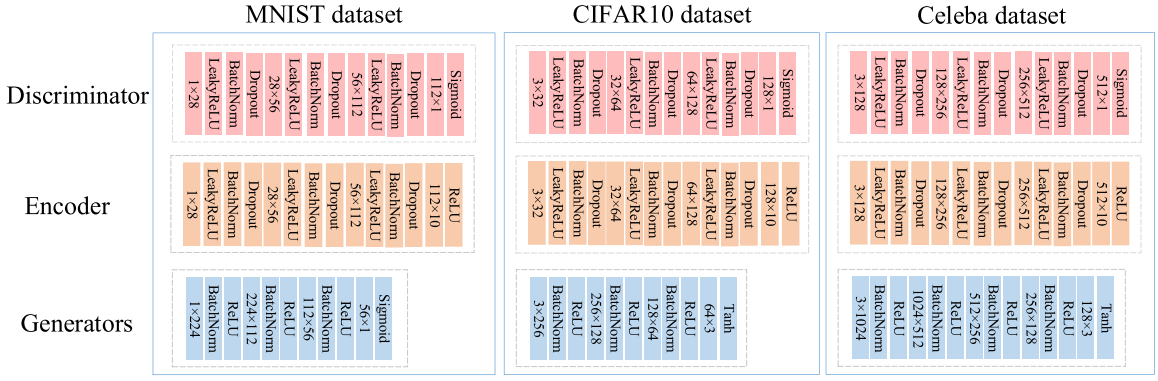


Fig. 4. Architectural details of MGO-GAN for MNIST, CIFAR10 and CelebA datasets. We use *Adam* gradient method [32] and *BCE* to update the parameters of all components. The parameter of LeakyRelu [29] is set to 0.02 and that of Dropout [33] is set to 0.5.

vector, we set the feature vector size to 10 in our experiments. Note that the relative performance of model is not sensitive to different feature vector sizes [31]. Besides, we conduct experiments on MNIST with feature vector size set to 20, the performance is similar to the case with feature vector size set to 10.

5.1. MNIST

MNIST dataset contains 10 categories from figure '0' to figure '9'. We first validate our proposed MGO-GAN on this dataset and compare experimental results with baselines. For MGAN and MGO-GAN, two generators are employed. Since MGO-GAN adopts orthogonal value to guide multiple generators to learn different information, and this is achieved by minimizing the orthogonal value along with minimizing the original generator loss, we illustrate the orthogonal values as shown in Fig. 5. Here, MGO-GAN produces the generated images at the nadir, which is marked by red ellipse. This is because smaller orthogonal value implicitly indicates that generators hold more different information. The generated images of MGO-GAN are shown in Fig. 6(e), and other sub figures ((a)–(d)) show the generated images of baselines.

The first five rows of sub-figure (d) and sub-figure (e) are from the first generator of MGAN and MGO-GAN, while the last five rows are from the second generator of both models. In Fig. 6, we can see that our proposed MGO-GAN outperforms other models. The generated images produced by WGAN lack figures '4', '5' and

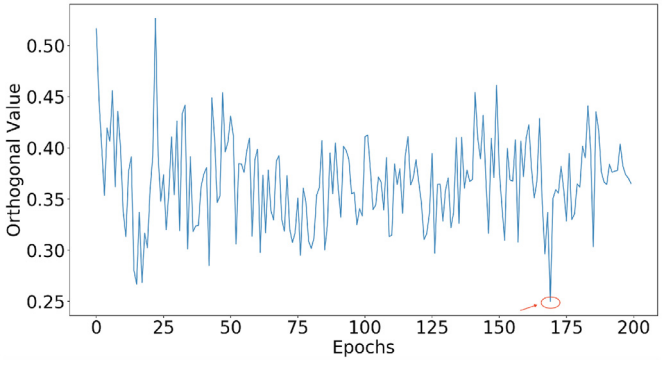


Fig. 5. The red ellipse indicates the minimal orthogonal value during training on MNIST dataset. At the nadir of the orthogonal value, MGO-GAN is finalized for generating synthetic images which are shown in Fig. 6(e). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

'6', while the samples generated by SN-GAN and LGAN hold low quality. As for MGAN, there exist many identical instances. In addition, MGAN does not capture the figures '0', '1', '2', '3' and '9'. This is because MGAN utilizes the shared parameters strategy to train model, which causes generators learning same modes. In MGO-GAN, all categories are captured with diverse synthetic instances.

(a) SN-GAN (b) WGAN (c) LGAN (d) MGAN **(e) MGO-GAN**

Fig. 6. Generated images by baselines and MGO-GAN. For MGAN and MGO-GAN, we employ 2 generators and each generator produces 50 samples in this case. The first five rows show the outputs of one generator and the last five rows show the outputs of another generator. As to SN-GAN, LGAN and WGAN, each of them produces 100 instances.

Table 1
MNIST scores and FID scores of generated data on MNIST dataset.

Model	MNIST Score	FID Score
WGAN [9]	6.006	75.806
SN-GAN [11]	6.514	61.153
LGAN [19]	6.760	83.278
MGAN [13]	7.176	124.427
MGO-GAN	**7.454**	**59.369**

Table 2
Inception scores and FID scores of generated data on CIFAR10.

Model	Inception Score	FID Score
WGAN [9]	4.035	218.195
SN-GAN [11]	3.905	229.135
LGAN [19]	2.896	293.714
MGAN [13]	5.087	225.499
MGO-GAN	**6.130**	**198.894**

The features (e.g., shape or angle) are different even within the same category. In addition, the experimental results also show that the generators of our proposed MGO-GAN learns different modes from each other. For generator one of MGO-GAN (the first five rows in sub-figure (e)), all figures are captured except for '5'; but the generated instances from the second generator (the last five rows in sub-figure (e)) do not hold the figure '2'. However, the combination of generated images from the two generators captures all modes of data, which shows the effectiveness of MGO-GAN on addressing the problem of mode collapse.

To quantitatively evaluate the quality and the diversity of generated images, we adopt the *MNIST Score* [34] and the *Frechet Inception Distance Score* (FID Score) [35]. MNIST Score is similar to *Inception Score* but using a classifier adapted to the MNIST training data instead of the Inception network [34]. For FID score, a lower value denotes more diverse generated images, and FID score becomes higher when generated instances are lack of diversity. This is because FID score is sensitive to mode collapse [35]. Assuming that original data distribution and generated data distribution are both multivariate Gaussian distributions, FID measures the Frechet distance, which is also the 2-Wasserstein distance, between the two distributions. The MNIST scores (the higher the better) and the FID scores (the lower the better) are shown in Table. 1. Obviously, our proposed MGO-GAN outperforms baselines in terms of achieving the highest MNIST score and the lowest FID score.

Since two generators capture few modes of data and produce many identical instances in MGAN, one may wonder whether the performance of MGAN could be improved by employing more generators while each generator is required to produce fewer instances. Thus, we conduct a pair of experiments in which the number of generators is set to 5 or 10, and each generator is only allowed to produce 10 instances for both MGAN and MGO-GAN. The generated images are shown in Fig. 7, and the samples at each row are generated by the same generator. Obviously, MGO-GAN still outperforms MGAN on both quality and diversity of generated samples. Specifically, the samples generated by MGAN contain many identical instances. In 5G case, G_3 and G_4 of MGAN learn the same category. In 10G case, G_2, G_6 and G_8, G_{10} of MGAN also learn the same category. Also, the synthetic images produced by one generator in MGAN are basically identical, showing poor diversity. For 10G case of MGO-GAN, G_1 tends to capture the mode '5'; G_2 tends to produce modes '1' and '2'; G_5 tends to generate mode '7'. Moreover, the generated figures within the same category present different angles and shapes, which shows the diversity. The union of all generated instances covers all categories. The same scenario also appears in 5G case. This is one more strong evidence to show the attractive property of MGO-GAN that multiple generators are able to capture diverse modes in a complementary manner due to the adoption of orthogonal vectors.

5.2. CIFAR10

We proceed to validate the baselines and our proposed MGO-GAN on CIFAR10 dataset. Similar to MNIST, MGAN and MGO-GAN adopt 2 generators to produce synthetic instances, and MGO-GAN still produces generated images at the nadir of orthogonal value during training. The orthogonal values are shown in Fig. 8, and the generated images for all models are shown in Fig. 9. For SN-GAN, the generated objects take the same color. This can also be seen as mode collapse. In this case, those images are slight hue into green. Since the generator of LGAN is made up of encoder-decoder, the generated images are blurred, holding the lowest quality among GAN variants. For MGAN, only part of modes in training data are captured while a lot of modes are missed. Inception scores [36] and FID scores achieved by MGO-GAN and other GAN variants are reported in Table. 2. The experimental results further confirm the significant performance improvement of MGO-GAN over baselines.

5.3. CelebA

Given that each image in CelebA dataset is rectangle, we reshape all images to the size $3 \times 128 \times 128$ for conveniently training networks. Similar to MNIST and CIFAR10 cases, we employ 2 generators for MGAN and MGO-GAN, and each generator produces 8 instances for convenient observation. Moreover, MGO-GAN still produces synthetic images at the nadir of orthogonal values which are shown in Fig. 10. The generated images for all models are shown in Fig. 11, and the FID scores are shown in Table 3. Still, our proposed MGO-GAN outperforms baselines. Although CelebA dataset has no label, it can still be differentiated with specific characteristics (e.g., rouging lips or not), and these characteristics can be used to demonstrate the diversity of generated data.

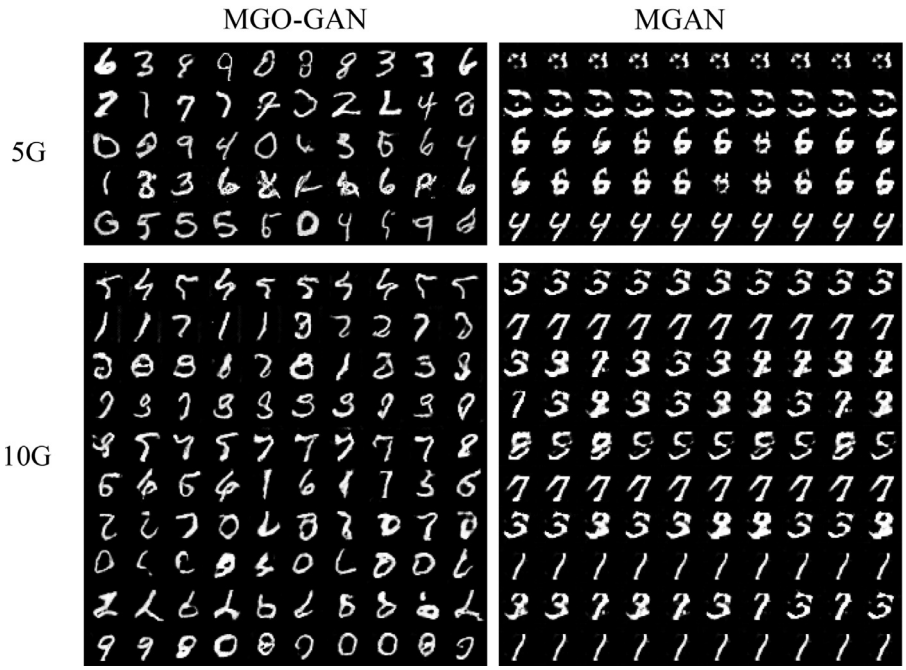


Fig. 7. Synthetic images produced by MGO-GAN and MGAN on MNIST. 5*G* (10*G*) indicates that 5 (10) generators are employed by MGAN and MGO-GAN. In 5*G* (10*G*), each row shows the outputs of one generator.

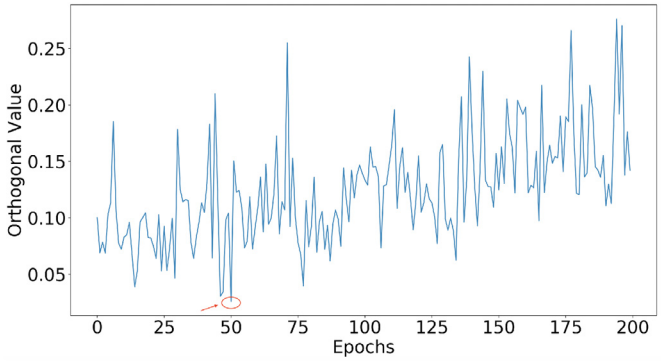


Fig. 8. The red ellipse indicates the minimal orthogonal value during training on CIFAR10 dataset. At the nadir of the orthogonal value, MGO-GAN produces synthetic images which are shown in Fig. 9(e). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

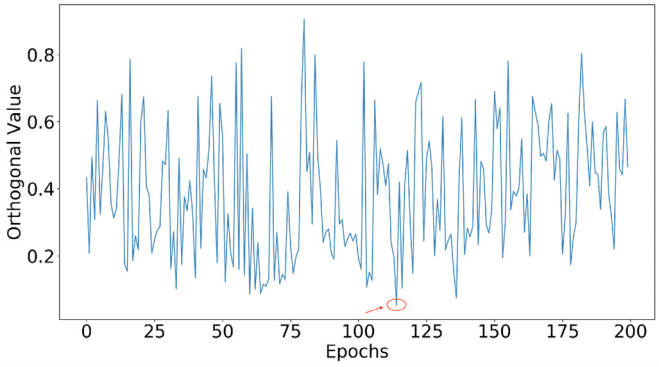
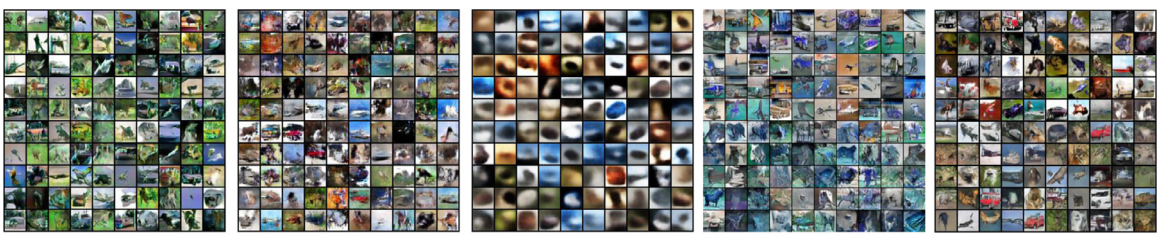


Fig. 10. The red ellipse indicates the minimal orthogonal value during training on CelebA dataset. At the minimal point, we select the generated image which is shown in Fig. 11(e). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

By comparing MGO-GAN with baselines, we can observe that the generated images produced by SN-GAN still fall into a specific color sub-space. In this case, those images take the gray color. LGAN seems like to show different styles of one object. In addition, the quality of generated images is not satisfactory. In MGO-GAN, the scenario in which different generators learn different information can be clearly observed. For instance, the first generator learned the features of a girl with lipstick (row 1, column 4 and row 2, column 3) while another generator learned the features of a girl with eye-shadow (row3, column4). This case also implicates the complementary manner playing an important part



(a) SN-GAN (b) WGAN (c) LGAN (d) MGAN **(e) MGO-GAN**

Fig. 9. Generated images by baselines and MGO-GAN. Similar to MNIST, the first five rows show the outputs of one generator and the last five rows show the outputs of another generator for MGAN and MGO-GAN.

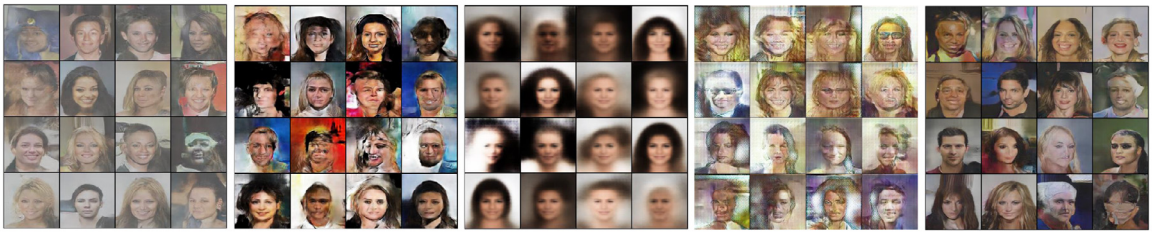


Fig. 11. Generated images by baselines and MGO-GAN. In this case, the first two rows show the outputs of one generator and the last two rows show the outputs of another generator for MGAN and MGO-GAN.

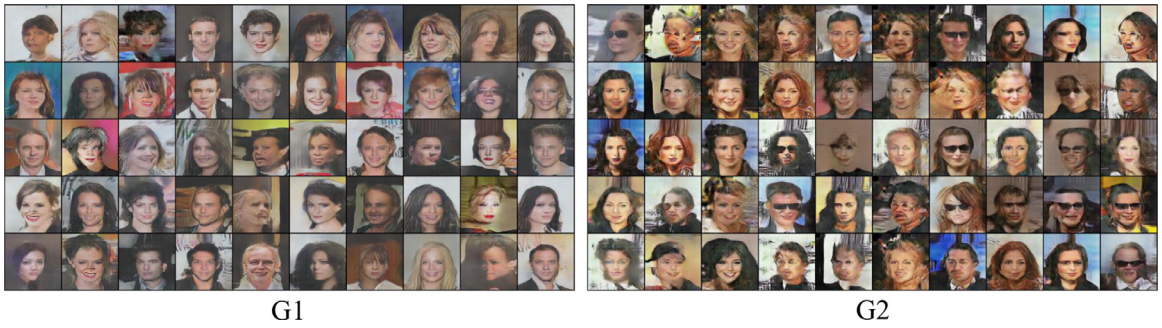


Fig. 12. The first generator $G1$ focuses on generating synthetic images with lipstick, while the second generator $G2$ basically produces images with glasses.

Table 3
FID scores of generated data on CelebA.

Model	FID Score
WGAN [9]	270.666
SN-GAN [11]	223.719
LGAN [19]	273.759
MGAN [13]	446.141
MGO-GAN	**189.188**

during training. As to WGAN, it does not loyally reflect diversity. For example, there is no one rouging his/her lips.

Given each generator in Fig. 11 only producing 8 images and the salient characteristics contained by generated images may not be supportive to show that different generators capture different information, here we encourage each generator producing 50 images in our proposed MGO-GAN. We utilize the statistics $\frac{object}{n}$ (here $n = 50$ and *object* refers to salient characteristics within an image) to count such generated images which contain salient characteristics (e.g., lipstick, eyeshadow, beard) captured by one generator while other generators do not hold. The generated images are shown in Fig. 12. From Fig. 12, we can observe that the first generator $G1$ focuses on generating synthetic faces with lipstick ($\frac{lipstick}{n}$ = 22%), while the second generator $G2$ basically produces faces with glasses ($\frac{glasses}{n}$ = 28%). Moreover, $G1$ generates the faces with double chin (row 3, column 1 and row 5, column 10, $\frac{dbould\ chin}{n}$ =4%) and with bangs (row 1, column 3 and column 5 as well as row 5, column 6, $\frac{bangs}{n}$ =6%); $G2$ generates the faces with eyeshadow (row 2, column 10 and row 4, column 8 as well as row 5, column 9, $\frac{eyeshadow}{n}$ =6%) and with beard (row 2, column 1, $\frac{beard}{n}$ =2%). This further demonstrates that the our proposed MGO-GAN can guide different generators capture different information in a complementary manner.

5.4. Discussion

The crucial point of employing multi-generator to address the problem of mode collapse is how to prevent generators learning the same modes. If there is no constraint, the performance of multi-generator would degenerate to the single generator case, such that the mode collapse still exists. In this paper, we propose a new approach to training multi-generator model to complementary learn different information of data with orthogonal vectors. The smaller orthogonal values reflect a lower correlation of information between two generators and a greater diversity of information between them. To minimize the correlation, we minimize the orthogonal value along with minimizing the original generator loss with back-propagation. Afterwards, we integrate the orthogonal value with the generator loss to update the corresponding generator's parameters. This results in a model which generates higher-quality and more diverse objects (See Fig. 6, Table 1, Fig. 7, Fig. 9, Table 2, Fig. 11, Table 3 and Fig. 12). In addition, our proposed model can avoid the case of $JSD(p_r||p_G) = \log 2$. This means that MGO-GAN can also address the vanishing gradient problem.

6. Conclusion

In this paper, to deal with the problem of mode collapse, we propose the MGO-GAN, which adopts multiple generators to learn different information of data with orthogonal vectors. To validate our proposed model, extensive experiments are conducted using MNIST, CIFAR10 and CelebA datasets. Both empirical and quantitative studies on generated images demonstrate the following capabilities of our proposed MGO-GAN. (*i*), MGO-GAN is capable of generating diverse and high quality instances at different resolutions (e.g., 28 $\times$ 28 for MNIST and 128 $\times$ 128 for CelebA). (*ii*), multiple generators of MGO-GAN are able to learn diverse modes in a complementary way. (*iii*), MGO-GAN outperforms other GAN variants in terms of achieving the highest MNIST score and Inception score and the lowest FID score.

For the success of MGO-GAN, one factor is the implicit assumption that throughout this work the training data modes are disconnected in the mapping space. Therefore, we would be interested in verifying this assumption in real-world application datasets, and studying the topological properties of these datasets in general in our future works. Moreover, it is worth to investigate how to make the discriminator being capable of assessing the diversity of generated data rather than only estimating the probability that current

samples were from training dataset or the generator, which would remain as exciting future paths for research in GAN community.

Declaration of Competing Interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled, "Tackling Mode Collapse in Multi-Generator GANs with Orthogonal Vectors".

Acknowledgment

This work was supported in part by the National Key R&D Program of China (No. 2018YFC1604000), and in part by the Science and technology project of Guangdong Provincial Tobacco Monopoly Administration (No. 2019440000200035).

Appendix A. MGAN becomes the vanilla GAN

$$\begin{aligned}
&\beta\left\{\sum_{k=1}^{K}\pi_k\mathbb{E}_{x\sim p_{G_k}}\log\frac{\pi_k p_{G_k}}{\sum_{j=1}^{K}\pi_j p_{G_j}}\right\}\\
&=\beta\left\{\sum_{k=1}^{K}\pi_k\mathbb{E}_{x\sim p_{G_k}}\log\frac{\pi_k p_{G_k}}{\sum_{j=1}^{K}\pi_j p_{G_j}}\right\}\\
&=\beta\left\{-\sum_{k=1}^{K}\pi_k\mathbb{H}(p_{G_k})+\mathbb{H}\left(\sum_{j=1}^{K}\pi_j p_{G_j}\right)+\sum_{k=1}^{K}\pi_k\log\pi_k\right\}
\end{aligned}$$

where $\mathbb{H}(p)$ indicates the Shannon entropy for distribution p, and the last term of this equation is a constant. If each generator in MGAN learns the homogeneous information, the value of $\mathbb{H}(p)$ is the same. In this way, the difference of first two terms is 0. Thus, $\mathcal{L}(G_{1:K})$ can be rewritten as:

$$\begin{aligned}
\mathcal{L}(G_{1:K}) &= \mathcal{J}(G,C^*,D^*)\\
&=\mathbb{E}_{x\sim p_{data}}\log\frac{p_{data}}{p_{data}+p_{model}}+\mathbb{E}_{x\sim p_{model}}\log\frac{p_{model}}{p_{data}+p_{model}}\\
&\quad-\beta\left\{\sum_{k=1}^{K}\pi_k\log\pi_k\right\}
\end{aligned}$$

Since $\pi_k\log\pi_k$ is a constant in MGAN, $\mathcal{L}(G_{1:K})$ is basically equivalent to the vanilla GAN loss function under such a scenario.

Appendix B. Proof of Eq. (7)

$$\begin{aligned}
V(G_{1:K}) &= \mathbb{E}_{x\sim p_r(x)}[\log D^*(x)]+\frac{1}{K}\sum_{i=1}^{K}\mathbb{E}_{z\sim p_z(z)}[\log(1-D^*(G_i(z)))]\\
&\quad+\frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)),E(G_j(z)))\\
&=E_{x\sim p_r}\log\frac{p_r}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}\\
&\quad+\frac{1}{K}E_{x\sim p_{G_1}}\log\frac{\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}\\
&\quad+\ldots+\frac{1}{K}E_{x\sim p_{G_K}}\log\frac{\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}\\
&\quad+\frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)),E(G_j(z)))\\
&=\int_x p_r\log\frac{p_r}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}dx\\
&\quad+\frac{1}{K}\int_x p_{G_1}\log\frac{\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}dx\\
&\quad+\ldots+\frac{1}{K}\int_x p_{G_K}\log\frac{\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}dx\\
&\quad+\frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)),E(G_j(z)))\\
&=\int_x p_r\log\frac{p_r}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}\\
&\quad+\frac{1}{K}p_{G_1}\log\frac{\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}\\
&\quad+\ldots+\frac{1}{K}p_{G_K}\log\frac{\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}dx\\
&\quad+\frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)),E(G_j(z)))\\
&=\int_x p_r\log\frac{p_r}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}\\
&\quad+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}\log\frac{\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}dx\\
&\quad+\frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)),E(G_j(z)))\\
&=\int_x p_r\log\frac{p_r}{\frac{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{2}}\\
&\quad+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}\log\frac{\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{\frac{p_r+\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K}}{2}}\\
&\quad-2\log 2+\frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)),E(G_j(z)))\\
&\approx -2\log 2+2JSD(p_r||\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K})\\
&\quad+\frac{1}{2}\sum_{i\neq j}\lambda\mathcal{O}(E(G_i(z)),E(G_j(z)))
\end{aligned}$$

where $\mathcal{O}(E(G_i(z)),E(G_j(z)))=\left|\frac{E(G_i(z))*E(G_j(z))}{|E(G_i(z))|*|E(G_j(z))|}\right|$, $i\neq j$ and $1\leq i\neq j\leq K$. $\mathcal{O}(E(G_i(z)),E(G_j(z)))$ is minimized along with minimizing the original generator loss, and this is achieved by training the criterion with back-propagation through $\nabla_\theta\left(\frac{1}{K}\log(1-D(G_{i_\theta}(z)))+\mathcal{O}(E(G_{i_\theta}(z)),E(G_{j_\theta}(z)))\right)$ in a similar way as vanilla GAN, $j\in 1{:}\ K$ and $j\neq i$. With the epochs increase, G_i gradually hold the information which other generators less hold, and this is implicitly reflected by $\mathcal{O}(E(G_i(z)),E(G_j(z)))$, $j\in 1:K$ and $j\neq i$, i.e., the value of $\mathcal{O}(E(G_i(z)),E(G_j(z)))$ becomes smaller. Assuming generator G_i finally holds such an information while other generators do not learn, the value of $\mathcal{O}(E(G_i(z)),E(G_j(z)))$ is smallest. Here, we term such a smallest value as δ. Thus, $V(G_{1:\ K})$ can be rewritten as:

$$V(G_{1:K})=-2\log 2+2JSD(p_r||\frac{p_{G_1}+p_{G_2}+\ldots+p_{G_K}}{K})+\delta$$

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, MIT Press, 2014, pp. 2672–2680.

[2] Q. Zhuang, K. HUANG, W. Qiu-Feng, X. Jimin, R. ZHANG, Generative adversarial classifier for handwriting characters super-resolution, Pattern Recognit. (2020) 107453.

[3] J. Gordon, J.M. Hernández-Lobato, Combining deep generative and discriminative models for bayesian semi-supervised learning, Pattern Recognit. 100 (2020) 107156.

[4] W. Chen, H. Hu, Generative attention adversarial classification network for unsupervised domain adaptation, Pattern Recognit. (2020) 107440.

[5] W. Li, W. Ding, R. Sadasivam, X. Cui, P. Chen, His-GAN: a histogram-based GAN model to improve data generation quality, Neural Netw. 119 (2019) 31–45.

[6] W. Li, L. Xu, Z. Liang, S. Wang, J. Cao, C. Ma, X. Cui, Sketch-then-edit generative adversarial network, Knowl. Based Syst. (2020) 106102.

[7] F. David, Generative Deep Learning: Teaching Machines to Paint, Write, Compose and Play, O'Reilly Media, 2019.

[8] A. Kumar, P. Sattigeri, T. Fletcher, Semi-supervised learning with GANs: Manifold invariance with improved inference, in: Advances in Neural Information Processing Systems, MIT Press, 2017, pp. 5534–5544.

[9] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein GAN, in: International Conference on Machine Learning, ACM, 2017.

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, Improved training of Wasserstein GANs, in: Advances in Neural Information Processing Systems, MIT Press, 2017, pp. 5767–5777.

[11] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, in: International Conference on Learning Representations, JMLR. org, 2018.

[12] A. Ghosh, V. Kulharia, V.P. Namboodiri, P.H. Torr, P.K. Dokania, Multi-agent diverse generative adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8513–8521.

[13] Q. Hoang, T.D. Nguyen, T. Le, D. Phung, MGAN: Training generative adversarial nets with multiple generators, in: International Conference on Learning Representations, JMLR. org, 2018.

[14] B.F. Green, The orthogonal approximation of an oblique structure in factor analysis, Psychometrika 17 (4) (1952) 429–440.

[15] J. Wang, J. You, Q. Li, Y. Xu, Orthogonal discriminant vector for face recognition across pose, Pattern Recognit. 45 (12) (2012) 4069–4079.

[16] P.W. Lamberti, A.P. Majtey, Non-logarithmic Jensen–Shannon divergence, Physica A 329 (1–2) (2003) 81–90.

[17] L. Rüschendorf, The Wasserstein distance and approximation theorems, Probab. Theory Relat. Fields 70 (1) (1985) 117–129.

[18] H. Gouk, E. Frank, B. Pfahringer, M.J. Cree, Regularisation of neural networks by enforcing Lipschitz continuity, Stat 1050 (2018) 14.

[19] G.-J. Qi, L. Zhang, H. Hu, M. Edraki, J. Wang, X.-S. Hua, Global versus localized generative adversarial nets, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1517–1525.

[20] I.O. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, B. Schölkopf, AdaGAN: boosting generative models, in: Advances in Neural Information Processing Systems, MIT Press, 2017, pp. 5424–5433.

[21] S. Arora, R. Ge, Y. Liang, T. Ma, Y. Zhang, Generalization and equilibrium in generative adversarial nets (GANs), in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, ACM, 2018, pp. 224–232.

[22] C.-C. Chang, C. Hubert Lin, C.-R. Lee, D.-C. Juan, W. Wei, H.-T. Chen, Escaping from collapsing modes in a constrained space, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 204–219.

[23] V. Eijkhout, P. Vassilevski, The role of the strengthened Cauchy–Buniakowskii–Schwarz inequality in multilevel methods, SIAM Rev. 33 (3) (1991) 405–419.

[24] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.

[25] B. Fuglede, F. Topsoe, Jensen-Shannon divergence and hilbert space embedding, in: International Symposium on Information Theory, 2004. ISIT 2004. Proceedings., IEEE, 2004, p. 31.

[26] M. Arjovsky, L. Bottou, Towards principled methods for training generative adversarial networks, in: International Conference on Learning Representations, JMLR. org, 2017.

[27] Y. Yu, Z. Gong, P. Zhong, J. Shan, Unsupervised representation learning with deep convolutional neural network for remote sensing images, in: International Conference on Image and Graphics, Springer, 2017, pp. 97–108.

[28] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How does batch normalization help optimization? in: Advances in Neural Information Processing Systems, MIT Press, 2018, pp. 2483–2493.

[29] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, in: International Conference on Machine Learning, ACM, 2015.

[30] M. Sion, et al., On general minimax theorems., Pac. J. Math. 8 (1) (1958) 171–176.

[31] D.P. Kingma, M. Welling, Auto-encoding variational bayes, in: International Conference on Learning Representations, JMLR. org, 2014.

[32] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: International Conference on Learning Representations, JMLR. org, 2014.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[34] C. Hardy, E. Le Merrer, B. Sericola, MD-GAN: Multi-discriminator generative adversarial networks for distributed datasets, in: 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2019, pp. 866–877.

[35] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, GANs trained by a two time-scale update rule converge to a local nash equilibrium, in: Advances in Neural Information Processing Systems, MIT Press, 2018, pp. 6626–6637.

[36] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training GANs, in: Advances in Neural Information Processing Systems, MIT Press, 2017, pp. 2234–2242.

Wei Li is now an Associate Professor of School of Artificial Intelligence and Computer Science, Jiangnan University. He received his 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.

Li Fan received Bachelor degree in computer science from LanZhou University in 2018. She is now a Master candidate in Wuhan University.

Zhenyu Wang received Ph.D. degree in computer science from Wuhan University in 2018. He is now a postdoc in Wuhan University. He studied at VIIT in India in 2009 and 2010. His current research interests include big data and blockchain with major applications in food safety and industrial data.

Dr. Chao Ma is currently an Assistant Professor of School of Cyber Science and Engineering at Wuhan University, P.R.China. His research interests include time series analytics, representation learning, social network analytics and big data. He has published over 20 academic papers in major international journals and conference proceedings. He is now the member of IEEE and the professional member of CCF.

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.