Knowledge-Based Systems 212 (2021) 106513

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Multi-generator GAN learning disconnected manifolds with mutual information

Wei Li [a,b,c], Zhixuan Liang [d], Julian Neuman [e], Jinlin Chen [d,*], Xiaohui Cui [f,*]

[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] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, PR China
[e] McKelvey School of Engineering, Washington University in St.Louis, MO, 63130, USA
[f] School of Cyber Science and Engineering, Wuhan University, Wuhan, Hubei, PR China

ARTICLE INFO

Article history:
Received 10 April 2020
Received in revised form 4 October 2020
Accepted 6 October 2020
Available online 10 November 2020

Keywords:
Mode collapse
GAN
Multiple generators
Mutual information

ABSTRACT

Original data usually lies on a set of disconnected manifolds rather than a smooth connected manifold. This causes the problem of mode collapse in the training of vanilla Generative Adversarial Network (GAN). There are many existing GAN variants that attempt to address this problem, but they result in limitations. The existing variants either produce simulated instances with low quality or generate identical simulated instances. In this study, we propose a new approach to training GAN utilizing multiple generators, a classifier and a discriminator to address mode collapse. The classifier outputs the statistical probabilities of generated data belonging to a specific category. These probabilities implicitly reflect which manifolds are captured by generators, and the correlation between generators is quantified by mutual information. Our idea views the mutual information values as a constraint to guide generators in learning different manifolds. Specifically, we traverse the generators, calculating the mutual information between each generator and the others. The calculated values are integrated into the generator loss to form a new generator loss and to update the corresponding generator's parameters, using back-propagation. We minimize the mutual information to reduce the correlation between generators while also minimizing the generator loss. This ensures generators capture different manifolds while updating their parameters. A new minimax formula is established to train our approach in a similar spirit to vanilla GAN. We term our approach *Mutual Information Multi-generator GAN* (MIM-GAN). We conduct extensive experiments utilizing the MNIST, CIFAR10 and CelebA datasets to demonstrate the significant performance improvement of MIM-GAN in both achieving the highest *Inception Scores* and producing diverse generated data at different resolutions.

1. Introduction

Generative Adversarial Network (GAN) [1] has been drawing much attention in the Deep Learning community due to its powerful generative capabilities [2–5]. A typical GAN model consists of two components, a discriminator D and a generator G. The discriminator D estimates a probability that a sample came from the training data rather than the generator G. The generator G specializes in generating simulated samples to fool D into accepting these samples as real. The GAN model sidesteps the difficulty of approximating many intractable probabilistic computations; it samples data from an easy-to-sample distribution so that Markov chains are unnecessary. Its gradients are tuned using back-propagation, which makes the training computationally inexpensive. Although GAN is successfully applied to many applications [6–8], training a GAN is still a challenge because it can easily fall into the problem of mode collapse.

Real datasets generally hold a set of disconnected manifolds [9]. However, the ReLU activation function [10] adopted by the generator G always outputs a continuous, piece-wise linear mapping [11]. In other words, the continuous generator seeks to find a discontinuous mapping in a specific space. Without ReLU activation, the GAN is unable to learn the non-linear information of the original data [12]. This intrinsic conflict renders the GAN model unable to converge or to converge to one continuous branch of the mapping, causing mode collapse [11]. The vanilla GAN has no strategy to prevent such a scenario. This is because the major task of G is to fool the discriminator into accepting the generated instances as real. D only outputs a probability that

* Corresponding authors.
E-mail addresses: cs_weili@jiangnan.edu.cn (W. Li), zhixuan.liang@connect.polyu.edu.hk (Z. Liang), jwneuman@wustl.edu (J. Neuman), csjlchen@comp.polyu.edu.hk (J. Chen), xcui@whu.edu.cn (X. Cui).

https://doi.org/10.1016/j.knosys.2020.106513
0950-7051/

estimates whether a sample is from the original dataset or not. Naturally, G can find success producing data only from specific manifolds, and there is no mechanism in place to prevent it from doing so.

Mode collapse in GAN indicates that: (1) only one or several manifolds of original data are learned — most of the manifolds are missing; (2) generated samples have identical instances. See Fig. 1 for a less formal, more pedagogical illustration of mode collapse. Many GAN variants have been developed to address this problem, however, they have respective limitations. Wasserstein GAN (WGAN) [13] adopts Wasserstein distance to measure the dissimilarity between the original data distribution (p_r) and the generated data distribution (p_G). However, the weight clipping method adopted by WGAN results in optimization difficulties and the inevitability of information loss, causing a capacity underuse issue [14]. Spectral Normalization GAN (SN-GAN) [15] adopts spectral normalization (λ) to make the discriminator Lipschitz continuous. This allows it to reserve the weight array (W) information to the greatest extent by using $\frac{W}{\sqrt{\lambda}}$. However, SN-GAN requires all layers of the discriminator to satisfy Lipshitz continuity. Such a strategy causes the generated objects to take the same color. This can be viewed as a manifestation of the mode collapse problem.

Recent works, motivated by the limitations of a single generator for learning multiple disconnected manifolds, attempt to employ multiple generators to learn all manifolds. Multiple Generators GAN (MGAN) [16] employs a mixture of generators and adopts a "shared parameters" strategy to simultaneously train all generators. Each time MGAN chooses a generator G_μ (μ indicates the generator id) and assumes μ follows multinomial distribution. Then, $G_\mu(z)$ is used as the output. The discriminator D aims to distinguish between this output and the training samples. However, the "shared parameters" strategy results in the tendency of generators to learn the same manifolds as each other. Under such a scenario, MGAN produces identical generated instances. The Disconnected Manifold Learning GAN with Prior Learning (DMGAN-PL) [9] model chooses to learn the prior ($p(c)$) over generators with an extra network (Q) rather than a fixed distribution (e.g., multinomial), and this model abandons the "shared parameters" strategy. Although prior learning is more suitable for fitting the distribution over generators, the process of prior learning is quite complicated and needs to consume a lot of computational resources. A less effectively trained network (Q) tends to cause DMGAN-PL model to produce synthetic instances with low quality. We can observe the details in the experiments section. Considering the significance of the challenges of the aforementioned models and the potential benefits of overcoming them, it is worth developing a new approach to address the mode collapse problem.

In this paper, we propose a new approach to training multi-generator GAN. Our approach is formulated as a minimax game among three parties: a set of generators $G_{1:n}$, a classifier C and a discriminator D. Generators $G_{1:n}$ produce synthetic instances that are intended to come from the same distribution as the training data, the discriminator D determines whether samples are true data or generated by generators, and the classifier C produces the statistical probabilities that generated data belongs to a specific category. The probabilities implicitly reflect which manifolds are captured by generators. As opposed to maximizing the correlation between generator id and generated instances as shown in the above multi-generator GAN variants, we minimize the correlation between generators with mutual information [17] to guide generators in learning different manifolds. Specifically, we travel over the generators and calculate the mutual information values between each generator and the others using statistical probabilities. The calculated values are integrated into the generator loss to form a new generator loss and utilized to update the corresponding generator's parameters with back-propagation. Since the mutual information consists of the correlation between each generator and the others, a generator's loss is implicitly aware of the information captured by other generators. We minimize the mutual information to reduce the correlation between generators, along with minimizing the generator loss. This helps generators capture different manifolds while updating their parameters. Mutual information values closer to 0 reflect a lower correlation of data between two generators and a greater diversity of information between them. In this way, we term our approach *Mutual Information Multi-generator GAN* **(MIM-GAN)**, and a detailed discussion is shown in Section 4.

In summary, the main contributions of this study are shown as follows:

- This paper proposes a new approach to training multi-generator GAN with mutual information, enforcing multiple generators learning different manifolds of data and overcoming the problem of mode collapse.
- This paper demonstrates how to learn different manifolds by minimizing mutual information between generators from both theoretical and empirical perspectives, providing new insight into the success of MIM-GAN.
- Through comprehensive experiments on three public datasets with different resolutions, we demonstrate the effectiveness of our proposed approach.

The rest of this paper is organized as follows. In Section 2 existing works are discussed. In Section 3 we review the GAN model and mutual information. In Section 4 we discuss our proposed MIM-GAN. In Section 5 we show our experimental results. Lastly, Section 6 serves as our conclusion.

2. Related work

Mode collapse is a major obstacle to GAN and its variants, and many studies have been proposed to tackle the mode collapse problem. Here, we categorize these studies according to the strategies they adopt.

Modifying Loss Function. Generator G learning one or several manifolds is a case in which the generated data distribution cannot be completely matched to the original data distribution. Such a distribution mismatch is a joint consequence of the negligible overlapping area and the JS divergence. JS divergence cannot measure the difference between p_r and p_G when the two distributions have negligible overlapping area. In this way, many researchers focus on modifying the loss function to address mode collapse.

Wasserstein GAN [13] is a classical model and it adopts the Wasserstein distance to measure the difference between p_r and p_G. In comparison to JS divergence, Wasserstein has a smoother value space and is better equipped to provide smooth representations of the distance in-between distributions in lower dimensional manifolds. [18]. For its approach to the Wasserstein distance, WGAN adopts the 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, $\frac{\partial f^*_w(x)}{\partial x}$ is limited to a specific range (e.g., $(-0.01, 0.01)$), which is known as weight clipping. The optimal strategy is either to take the largest value (i.e., 0.01) or to take the smallest value (i.e., -0.01) for all parameters under such a scenario. Weight clipping can cause gradients to vanish when the window is too small and weight clipping can slow convergence when the window is too

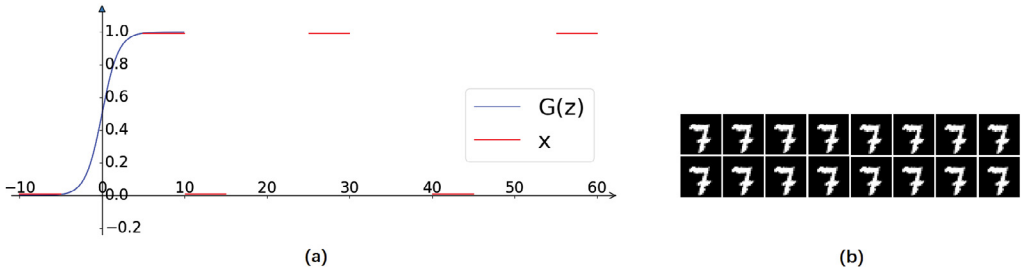


Fig. 1. Illustrative example of continuous representation from latent space $\mathcal{Z}$ to original data space χ. The generator $G(z) : \mathcal{Z} \rightarrow \chi$ with prior $z \sim$ Gaussian(0, 1) expects to learn all manifolds of original data. Sub-figure (a) shows an example of learning a part of the manifolds, in which the red line indicates the manifolds while the blue curve indicates the learned manifolds by G. Generated results are shown in sub-figure (b) and they are identical.

large. Moreover, only certain optimizers (e.g., *RMSProp* or *SGD*) in WGAN are suitable for optimizing WGAN [13], and momentum based optimizers (e.g., *Adam*) may turn the gradients negative.

Takeru et al. [15] proposed a novel weight normalization method to achieve the 1-Lipshitz continuity, termed spectral normalization GAN (SN-GAN). SN-GAN uses this spectral normalization to control the Lipschitz constant of the discriminator function f. For calculating the spectral normalization, SN-GAN adopts Singular Value Decomposition (SVD) to get the largest singular value and views this value as the spectral normalization. SN-GAN 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 penalizing the spectral norm [19]. Note that although the term $\frac{W_{ij}}{\sqrt{\lambda_1}}$ helps model achieve 1-Lipshitz continuity, it, in turn, limits the diversity of generated data, resulting in all generated objects taking the same color. This case can also be viewed as mode collapse.

Least Squares GAN (LSGAN) [20] argues that the Sigmoid Cross Entropy loss function [21] is not suitable for training the discriminator. Therefore, LSGAN uses Least Squares to measure the difference between p_r and p_G rather than Sigmoid Cross Entropy. The Least Squares measurement metric, however, still encounters a challenge when one of p_G or $p_r \rightarrow 0$ and the other one $\neq 0$ as both components (discriminator D and generator G) reach the optimal state. In this case, the loss function would be a constant (i.e., 1), and the constant cannot accurately reflect the real difference between p_r and p_G. The recent DMGAN [4] utilizes the combination of identity preserving loss and discriminative loss to generate synthetic samples with high quality. Still, the *Inception* score of our proposed MIM-GAN (6.233) is higher than that of DMGAN (6.08). As for MS^{2}GAN [5], it focuses on cross-modal retrieval.

Modifying 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 is not focused on all of the manifolds. An alternative approach to address this problem is to increase the quantity of generators in an attempt to learn more about different manifolds. Tolstikhin et al. [22] 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 *Choose Mixture Weight* and the *Update Training Weight* functions for boosting techniques. Arora et al. [23], alternatively, trained a mixture of a generator 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. [24] 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.

Hoang et al. [16] employed many generators $G_{1:K}$ in an effort to overcome the mode collapse problem, and they named the method MGAN. The K generators together induce a mixture over K distributions p_{model} in a specific space. $G_\mu(z)$ is the output in which μ is the index and it follows a multinomial distribution. The goal of MGAN is to minimize the JS-divergence between p_{model} and p_r while maximizing the same divergence between each generator, with shared parameters. An extra classifier C performs a multi-class classification to determine whether the generated samples, labeled by the index μ, are from the corresponding generator. Although study [16] claims 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 within the MGAN shared parameters method. Despite the index (i or j in MGAN) following multinomial distribution and elements within multinomial distribution being mutually exclusive, G_i and G_j are not necessarily mutually exclusive. Therefore, MGAN cannot guarantee the elimination of mode collapse. Additionally, MGAN does not prove that the modified loss function converges to an equilibrium. These issues cause mode collapse in MGAN to be more serious.

Instead of fixing the distribution over generators, Mahyar Khayatkhoei et al. [9] proposed Disconnected Manifold Learning GAN with Prior Learning (DMGAN-PL). The DMGAN-PL model utilizes an encoder network (Q) to learn the index distribution ($p(c)$) over generators. As suggested by study [17], DMGAN-PL achieves this by maximizing the mutual information $\mathcal{I}(c; x)$ between generator id (c) and generated instances (x), because DMGAN-PL thinks that each generated sample should be a perfect indicator of which generator it came from. However, we cannot guarantee that the maximal mutual information is obtained if each time the generated instances do not belong to generator G_c. Under such a scenario, the generated instances have no relation (or less relation) to generator id, such that the generators may still capture the same manifolds. In addition, learning $p(c)$ with a network is quite complicated and consumes undue computational resources. A less effectively trained network results in the production of synthetic instances with low quality, given that x follows $p_G(x|c)$. Based on these challenges, the DMGAN-PL authors stress that their approach is not an attempt to achieve optimal performance, but it is instead meant to highlight the effectiveness of their approach for learning prior [9]. With the

DMGAN-PL model there exists room for improved methodologies in addressing the mode collapse problem. This paper proposes a new approach to training multi-generator GAN.

Next, we discuss how to guide generators in learning the disconnected manifolds with our proposed approach.

3. Preliminaries

3.1. Generative Adversarial Network

Although Generative Adversarial Network (GAN) [1] was introduced in the first section, we formally describe it as follows to establish continuity. GAN was developed by Goodfellow as a novel generative model to simultaneously train a generator G and a discriminator D using the following function:

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

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). The generator G outputs realistic instances to fool the discriminator D into accepting the instances as real by maximizing its score $D(G(z))$. This is achieved by the following optimization function.

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

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.

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

In this work, the discriminator and the generator are alternatively optimized, and Jensen–Shannon (JS) divergence is utilized to measure the difference between the original data distribution and the generated data distribution. JS divergence reaches its lowest value as the discriminator and the generator reach a Nash equilibrium [25] where $D(G(z)) = D(x) = 0.5$. The GAN model converges under such a scenario.

3.2. Mode collapse

Theoretically, the problem of mode collapse does not easily happen. The algorithm of vanilla GAN [1] shows that the training process is based on a stochastic strategy, i.e., a stochastic gradient descent optimization or a stochastic sampling method ($z \sim p_z(z)$ and $x \sim p_r(x)$). This means that the gradients backpropagated to the generator G will be different at each epoch. Given that gradients have an influence on the generated instances, each time G gets different gradients, G should produce different instances. This analysis is, however, only a theoretical one. The GAN model may reach a near-equilibrium, when training succeeds in GAN, which happens from certain hyperparameter combinations [26]. Mode collapse still often occurs in practice.

Assuming the original data is supported on the manifold Λ, which has a set of disconnected manifolds, each indicated by M_i, we get $\Lambda = \bigcup_{i=1}^{n} M_i$. The prior $p_z(z)$ is often a bounded distribution (e.g., $\mathcal{N}(0,1)$ or $\mathcal{U}(-1,1)$), and is contained in the random space $\mathcal{Z}$. In the process of training a GAN, G is designed to build a mapping function which maps noise from the randomized space $\mathcal{Z}$ into data space χ. However, such a mapping is not surjective [27], such that just a part of the original data manifolds ($\bigcup_{i=1}^{k} M_i, k < n$) are learned by G. In this way, D cannot distinguish the generated data from real data, because D is only used to evaluate the probability that a sample is from the original dataset rather than generated by G. Although this generator produces realistic-like data, the remaining manifolds are missed under such a scenario.

We assume that D always maximizes the probability of assigning the correct label to both the original data and samples produced by G (See Eq. (3)). With this in place, D judges all samples generated by G as fake (low probability for $D(G(z))$) even though G has learned some original data manifolds. To fool D, G searches for other manifolds to learn. Since the mapping is not surjective, GAN repeats such a cat-and-mouse game and becomes stuck. Note that there is no countermeasure in Eq. (1) that explicitly forces generator G to escape 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.

3.3. Mutual information

Mutual Information [17] is a metric of the mutual dependence between two random variables. It can quantify the amount of information obtained from one random variable through observing another random variable. In this way, mutual information can objectively reflect the correlation between the two variables. The definition of mutual information is shown as follows.

Given a pair of random variables, (X, Y), over the space $\mathcal{X}\times\mathcal{Y}$. Their joint density function is $p(x,y)$, and the marginal probability density functions are $p(x)$ and $p(y)$ respectively. Mutual information can be defined as follows.

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)log\frac{p(x,y)}{p(x)\times p(y)} \tag{4}$$

We now take the MNIST dataset to discuss the mutual information in MIM-GAN. Assuming there are two generators G_1 and G_2, each generator produces 100 generated images at each epoch. C performs 10-class classification and it has enough capacity. Each generator attempts to learn figures '0'-'9'. After feeding all the generated images into the classifier C, C always accurately predicts the labels of generated data, and the probability of $G_1(z)$ ($G_2(z)$) is termed $p(G_1)$ ($p(G_2)$) in our study. For example, G_1 has learned the figures '0'-'4' and each figure holds 20 instances. $p(G_1(z)=0) = p(G_1(z)=1) = p(G_1(z)=2) = p(G_1(z)=3) = p(G_1(z)=4) = \frac{1}{5}$. $p(G_1)$ indicates the marginal probability density function $p(x)$ of variable X whilst $p(G_2)$ indicates the marginal probability density function $p(y)$ of another variable Y.

We consider a case in which G_1 has learned the figures '0'-'4' whilst G_2 has also learned the figures '0'-'4' at a specific moment. Each figure still holds 20 generated images, for convenient discussion. Under such a case, the mutual information between G_1 and G_2 is higher than the case where G_2 has, instead, learned the figures '5'-'9'. In the later case, the correlation between the two models is lower than in the former, resulting in lower mutual information as well. Hence, MIM-GAN needs to minimize the correlation between generators to make generators learn different information, and this can be achieved subsequently with the minimization of the original generator loss.

In the next section, we present our proposed MIM-GAN and give a theoretical analysis, essentially showing that the training criteria allows MIM-GAN to learn a set of disconnect manifolds.

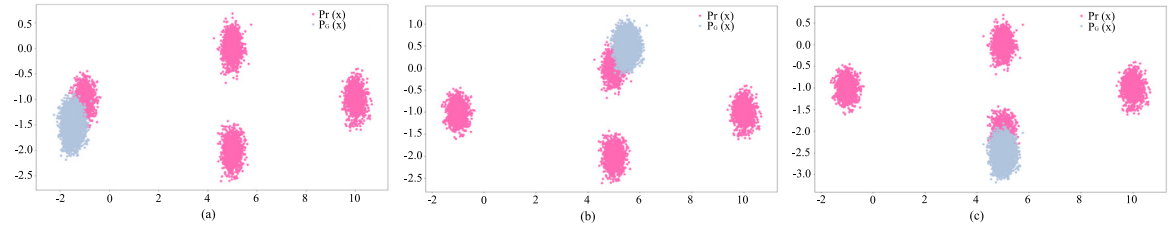


Fig. 2. $P_r(x)$ is the four-Gaussian (pink clusters) and we assume in this case each Gaussian indicating a single manifold. What we expect is to map a given random Gaussian into the four-Gaussian with G. However, the example shows that G just maps the random Gaussian to a certain Gaussian at different steps (steel-blue cluster in sub-figures (a), (b) and (c)).

4. MIM-GAN

To address the problem of mode collapse, MIM-GAN employs mutual information to ensure that multiple generators learn disconnected manifolds. To this end, there are two important issues to be addressed: (1) How the generators learn the disconnected manifolds by minimizing mutual information; and (2) Will MIM-GAN converge to the equilibrium.

4.1. Disconnected manifolds learning

We first introduce a set of generators $G_{1:n}$ to build a mapping function that can map a random noise code z from randomized space $\mathcal{Z}$ into the data space χ. We define a prior on $\mathcal{Z}$ as $p_z(z)$, where the generators sample noise from. The original data follows a specific distribution which we term $p_r(x)$. As a GAN variant, MIM-GAN still learns with the minimax game, i.e., maximizing the discriminator D and minimizing the generators $G_{1:n}$. As opposed to the vanilla GAN model, each generator in MIM-GAN is encouraged to focus on different manifolds of the original data. In this way, the objective function of MIM-GAN can be defined as follows:

$$\begin{aligned}\min_{G_{1:n}}\max_{D}\ V(G_{1:n},D) &= E_{x\sim p_r(x)}logD(x)\\ &+\frac{1}{n}\sum_{k=1}^{n}E_{z\sim p_z(z)}[log(1-D(G_k(z)))]\\ &+\frac{1}{2}\sum_{i\neq j}\lambda I(p(G_i(z));p(G_j(z)))\end{aligned}\tag{5}$$

where n indicates the number of generators and λ indicates a coefficient. The classifier C performs multi-class classification to produce statistical probabilities of generated data for mutual information calculation, and those probabilities produced by the ith generator are termed $p(G_i(z))$. Since D still tries to distinguish the generated data from real data by minimizing $D(G_{1:n}(z))$ and maximizing $D(x)$, and $G_{1:n}$ attempts to fool D into accepting their outputs as real data by maximizing its score $D(G_{1:n}(z))$, the generators and the discriminator are alternatively optimized by the following functions:

$$\begin{aligned}\min_{G_{1:n}}\ V_{MIM-GAN}(G) &= \frac{1}{n}\sum_{k=1}^{n}E_{z\sim p_z(z)}[log(1-D(G_k(z)))]\\ &+\frac{1}{2}\sum_{i\neq j}\lambda I(D(G_i(z));D(G_j(z)))\end{aligned}$$

$$\begin{aligned}\max_{D}\ V_{MIM-GAN}(D) &= E_{x\sim p_r(x)}logD(x)\\ &+\frac{1}{n}\sum_{k=1}^{n}E_{z\sim p_z(z)}[log(1-D(G_k(z)))]\end{aligned}$$

Assuming the discriminator D has enough capacity, we show below that the optimal discriminator D is at the equilibrium point $p_r = \frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}$, given that D is only used to estimate a probability that a sample is from the original dataset rather than generated by generators.

Proposition 1. *For generators G_1, G_2, …, G_n fixed, the optimal discriminator D is*

$$D^*(x) = \frac{p_r}{p_r + \frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}\tag{6}$$

Proof. The training criteria for the discriminator D, given a mixture of generators $G_1, G_2, \ldots, G_n$, is to maximize the quantity $V(D)$. Note that all generators $G_{1:n}$ sample noise code from a single prior distribution $p_z(z)$. $p_z(z)$ is the total probability in MIM-GAN, and $\mathcal{P}(G_i)$ is the probability of choosing G_i. Hence, $p_z(z) = \sum_{i=1}^{n}\mathcal{P}(G_i)\times p(z|G_i)$. Here, we assume each generator G_i holds the same sampling proportion. In other words, $\mathcal{P}(G_1) = \mathcal{P}(G_2) = \cdots = \mathcal{P}(G_n)$. For example, we assume $Z = z_1, z_2, \ldots, z_{100}$ and $n = 10$, generator G_1 samples noises $z_1, z_2, \ldots, z_{10}$ and $z_{11}, z_{12}, \ldots, z_{20}$ are allotted to generator G_2, and so on. In this way, we get:

$$\begin{aligned}V(D) &= \int_x p_r(x)logD(x)dx\\ &+\frac{1}{n}\sum_{k=1}^{n}\int_z p_z(z|G_k)\mathcal{P}(G_k)log[1-D(G_1(z))]dz\\ &=\int_x p_r(x)logD(x)+\frac{1}{n}p_{G_1}(x)log[1-D(x)]\\ &+\frac{1}{n}p_{G_2}(x)log[1-D(x)]+\cdots\\ &+\frac{1}{n}p_{G_n}(x)log[1-D(x)]dx\\ &=\int_x p_r(x)logD(x)\\ &+\frac{p_{G_1(x)}+\cdots+p_{G_n(x)}}{n}log[1-D(x)]dx\end{aligned}$$

It can be seen that D achieves its maximum in [0,1] at $p_r = \frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}$. This is because each generator G_i forms a single distribution p_{G_i}, n generators together induce a mixture of an empirical distribution, which is used to fit p_r. We compute the partial derivation of $\frac{\partial V(D(x))}{\partial D(x)}$, and we get $\frac{p_r}{D(x)} = \frac{1}{n}\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{1-D(x)}$. In this way, Eq. (6) is obtained, and the maximum of Eq. (6) is $\frac{1}{2}$ when $p_r = \frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}$. This implicitly indicates that the discriminator cannot distinguish the mixture generated data distribution from the original data distribution. Based on Proposition 1, we substitute Eq. (6) into Eq. (5), and we get:

$$V(G) = E_{x\sim p_r}log\frac{p_r}{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}$$

$$
\begin{aligned}
&+\frac{1}{n}E_{x\sim p_{G_1}}log\frac{\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}+\cdots\\
&+\frac{1}{n}E_{x\sim p_{G_n}}log\frac{\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}\\
&+\frac{1}{2}\sum_{i\neq j}\lambda I(p(G_i(z));p(G_j(z)))\\
=&\int_x p_r log\frac{p_r}{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}dx\\
&+\frac{1}{n}\int_x p_{G_1}log\frac{\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}dx\\
&+\cdots+\frac{1}{n}\int_x p_{G_n}log\frac{\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}dx\\
&+\frac{1}{2}\sum_{i\neq j}\lambda I(p(G_i(z));p(G_j(z)))\\
=&\int_x p_r(x)log\frac{p_r}{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}\\
&+\frac{p_{G_1(x)}+\cdots+p_{G_n(x)}}{n}log\frac{\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}dx\\
&+\frac{1}{2}\sum_{i\neq j}\lambda I(p(G_i(z));p(G_j(z)))
\end{aligned}
$$

Theorem 1. *The global minimum of the virtual training criterion $V(G)$ is achieved if and only if $\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}=p_r$ and each generator captures manifolds which other generators do not capture. Under such a scenario, the score of mutual information is small enough and it is assumed as δ in our study. Hence, $V(G)$ achieves the optimal value $-2log2+\delta$ at that point.*

Proof. For $\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}=p_r$, $D^*(x)=\frac{1}{2}$. In this way, we can observe that $V(G)$ reaches the value $-2log2+\delta$. This is because $\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}$ is viewed as a mixture distribution of generated data induced by $G_{1:n}$. Assuming the classifier C has enough capacity and it can accurately predict the labels of generated data at each epoch, $p(G_i)$ then royally reflects the statistical probabilities of generated data which belong to a specific category. In this way, the mutual information $I(p(G_i);p(G_j))$ represents the correlation between G_i and G_j. We minimize the correlation between generators to guide generators learning different manifolds. Assuming each generator holds manifolds which other generators do not hold, the correlation between any two generators is small enough. Hence, we obtain:

$$
\begin{aligned}
V(G)=&-2log2+\int_x p_r log\frac{p_r}{\frac{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{2}}\times dx\\
&+\int_x(\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n})log\frac{\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{\frac{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{2}}dx\\
&+I(p(G_i(z));p(G_j(z)))\\
=&-2log2+KL(p_r\parallel\frac{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{2})\\
&+KL(\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}\parallel\frac{p_r+\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}}{2})\\
&+I(p(G_i(z));p(G_j(z)))\\
=&-2log2+2JSD(p_r\parallel\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n})\\
&+I(p(G_i(z));p(G_j(z)))
\end{aligned}
$$

where KL is the Kullback–Leibler divergence [28] and JSD indicates Jensen–Shannon divergence [29]. The generated data produced by all generators fools the discriminator into accepting it as real data when $p_r=\frac{p_{G_1}+p_{G_2}+\cdots+p_{G_n}}{n}$. Under such a scenario, JSD holds a minimum of 0 and $I(p(G_i(z));p(G_j(z)))$ also holds a minimum of δ. Here, we discuss the term $I(p(G_i(z));p(G_j(z)))$.

$$
\begin{aligned}
&I(p(G_i(z));p(G_j(z)))\\
&=H(p(G_i(z)))+H(p(G_j(z)))-H(p(G_i(z)),p(G_j(z)))\\
&=-\sum p_i logp_i-\sum p_j logp_j+\sum p_{(i,j)}logp_{(i,j)}
\end{aligned}
$$

where $p(G_i(z))$ ($p(G_j(z))$) indicates the statistical probabilities of generated instances from ith (jth) generator, and this is calculated by a trained classifier. Note that $I(p(G_i(z));p(G_j(z)))$ contains the information learned by the ith and jth generators. When we travel over each generator but the ith one ($j\in 1:n$ and $j\neq i$), the mutual information values are aware of the information learned by other generators. Based on this, we can guide generators toward holding different manifolds through minimizing the mutual information along with minimizing generator loss. Assuming θ indicates the parameters of the ith generator, the minimizing process is achieved by training criterion with back-propagation through

$$
\theta_{G_i}\leftarrow\nabla_\theta\left(\frac{1}{K}log(1-D(G_i(z;\theta)))+\sum_{i\neq j}I(C(G_i(z;\theta)),C(G_j(z)))\right)
$$

where $j\in 1:n$ and $j\neq i$. Note that we fix other networks when we update the parameters of G_i, and other generators also adopt the same strategy to update their parameters as G_i. If $V(G)$ reaches the global minimum, the mutual information also reaches its minimum. Hence, the global optimal $V(G)=-2log2+\delta$.

Theorem 2. *The probability of the occurrence of gradients vanishing is significantly reduced, and its probability approaches 0 if n is enough large.*

Proof. For a single generator, the original data distribution p_r and the generated data distribution p_G have negligible, or even zero, overlapping area with a large probability. Assume this probability is $\mathcal{P}$. MIM-GAN employs multiple generators ($G_{1:n}$) and guides these generators to learn different manifolds of data. In this way, the probability is reduced to $\frac{\mathcal{P}}{n}$. If $n\rightarrow\infty$, the probability $\mathcal{P}$ approximates 0.

The pseudo-code of the algorithm is formally presented in **Algorithm 1**. Moreover, our generator and discriminator architectures are from vanilla DCGAN [30], and the details of our adopted components are still $Conv-BatchNorm-ReLu$ [31] (Generator G) or $Conv-BatchNorm-LeakyReLu$ [32] (Discriminator D), which follows the recent multi-generator GAN models (e.g., MGAN [16], DMGAN-PL [9]). This is because we do not need to enforce the discriminator 1-Lipschitz continuity [33, 34], and as such is the case, batch normalization is necessary in the MIM-GAN discriminator to scale the variances of various learning features into a consistent range. As for the classifier C, we adopt *Inception Network* (i.e., Inception-V4). In **Algorithm 1**, I_{idx} indicates the mutual information value among the current generator (G_{idx}) and other generators ($G_i, i\neq idx$ and $i\in[1,n]$), i.e., $I(C(G_{idx}(z));C(G_1(z)))$, …, $I(C(G_{idx}(z));C(G_{idx-1}(z)))$, $I(C(G_{idx}(z));C(G_{idx+1}(z)))$, …, $I(C(G_{idx}(z));C(G_n(z)))$. After that, we integrate the mutual information value I_{idx} with the original generator loss to form a new loss and utilize it to update the parameters of the corresponding generator (G_{idx}) with back-propagation.

Algorithm 1 MIM-GAN.

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

Fun: Cal-MI(idx, n):
rst $\leftarrow 0$
$G_{idx}(z)$ is fed into C to obtain $p(G_{idx})$.
for 1:n **do**
$G_i(z)$ is fed into C to obtain $p(G_i)$.
rst += $I(p(G_i), p(G_{idx}))$, $idx \neq i$
return rst

Adam optimizer and *BCE* loss function and n generators
for number of iterations **do**

- Sampling minibatch of $n \times m$ noise samples $z^1, \ldots, z^{n\times m}$ from $p_z(z)$, Gaussian distribution with (0,1). Each generator holds m noise samples.
- Sampling minibatch of m original samples $x^1, \ldots x^m$ from real dataset.
- Updating the discriminators parameters by ascending its stochastic gradient.
- $\nabla_{\theta_D} \frac{1}{m} \sum_1^m \{logD(x^{(i)}) + \frac{1}{n} log(1 - D(G_{1:n}(z^{(i)})))\}$.
- Sampling minibatch of $n \times m$ noise samples $z^1, \ldots, z^{n\times m}$.
- Iteratively calculating the mutual information I_{idx}=Cal-MI(idx,n) on each generator (G_{idx}), $idx \in [1:n]$.
- Updating the generators' parameters by descending its stochastic gradient.
- $\nabla_{\theta_{G_{idx}}} \frac{1}{m} \sum_1^m \{\frac{1}{K} log(1 - D(G_{idx_\theta}(z^{(i)})) + I(C(G_{idx_\theta}(z^{(i)})), C(G_j(z^{(i)})))\}$, $j \in 1:n$ and $j \neq idx$.

end for

The other generators also adopt the same strategy to update their parameters. The architecture of our proposed MIM-GAN is shown in Fig. 3.

4.2. Convergence of Algorithm 1

Proposition 2. *Assuming the discriminator D is in the optimal status at each step of* ***Algorithm 1*** *and holds enough capability, given* $G_{1:n}$, $\sum_i^n p_{G_i}$ *converge to* p_r.

Proof. Considering an optimal status, each generator only learns a specific manifold and $U(p_{G_{1:n}}, D)$ is convex in $p_{G_{1:n}}$. Let $p_{G_{1:n}} \times D \rightarrow V(G_{1:n}, D)$ be convex in its first argument and concave in its second argument. The supremum of this convex function is at the point where the maximum for optimal D is attained according to the Sion minimax theorem [35]. In other words, $SUP_D(p_{G_{1:n}}, D)$ is convex in $p_{G_{1:n}}$ with a unique global optima. Therefore, $p_{G_{1:n}}$ converges to p_r.

5. Experiments

For the experiments, the implementation details of MIM-GAN are shown in Fig. 4. Note that each generator in MIM-GAN is initialized by the same hyperparameters (Normal(0.0, 0.02) and biases are 0.0) and samples noise from the standard Gaussian distribution (0, 1) ($p_z(z)$). Five recently popular GAN variants are employed as baselines, which are **WGAN** [13] **SN-GAN** [15], **MGAN** [16], **RaGAN** [36] and **DMGAN-PL** [9]. They are representative of attempts to address mode collapse.

To make a fair comparison, WGAN and SN-GAN will also sample the same noise distribution ($\mathcal{N}(0, 1)$) while vanilla MGAN and DMGAN-PL sample the isotropic multivariate Gaussian distribution $\mathcal{N}(0, \mathbf{I})$. As for other model details, we use experimental settings that are identical to previous works as baselines. The training epoch in our study has been set the same for all models (i.e., $Epoch = 200$), and the learning rate is set to 0.0002 for MGAN, SN-GAN, DMGAN-PL and MIM-GAN while 0.0001 for RaGAN and 0.00005 for WGAN.

Three commonly used public image datasets, the MNIST, CIFAR-10 and Celeba, are studied. The MNIST dataset contains 50000 gray-scale images with size 28×28. Both the CIFAR-10 dataset and the Celeba dataset contain RGB images. The number of images in the CIFAR-10 dataset is 50000 and the size is $3 \times 32 \times 32$. The number of images in the Celeba dataset is 202599 and the size is $3 \times 178 \times 218$. The MNIST and the CIFAR-10 datasets are labeled and both consist of 10 categories. The Celeba dataset is a face dataset, and it has no explicit categories because it solely contains the face images. Nevertheless, the Celeba dataset can be labeled with specific characteristics (e.g., wearing a hat or not, with glasses or not), and these characteristics can be used to intrinsically group images into different categories and show the diversity of generated data.

5.1. MNIST dataset

We first apply MIM-GAN and baselines to MNIST dataset, and the 10 categories are '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', respectively. For WGAN and SN-GAN, we directly generate 100 instances. For MGAN, DMGAN-PL and MIM-GAN, the number of generators is set to 10, and each generator produces only 10 instances. The generated results are shown in Fig. 5, and the generated samples at each row in sub-figure (c), sub-figure (d) and sub-figure (e) are from the same generator. For example, the generated images in row 1 are from G_1.

In Fig. 5, we can see that our proposed MIM-GAN outperforms other models. The generated data produced by WGAN lacks the figures '4', '5' and '6', and the quality of generated data is the worst. The samples generated by MGAN contain many identical instances. Generator 2, generator 7, generator 8 and generator 10 in MGAN learn the same manifolds. This is because MGAN utilizes the shared parameters strategy among its generators. The generators hold the same parameters during training and consequently generate identical samples. In addition, MGAN does not learn the figures '0' and '2'. The results prove that MGAN cannot address the problem of mode collapse. As for SN-GAN and DMGAN-PL, the quality of their generated data is not satisfactory, which is shown in the subsequent Table 1. MIM-GAN, alternatively, learns the manifolds of data in a mutually exclusive manner. For example, in sub-figure (d), G_2 tends to produce figure '9' while G_7 tends to produce figure '4'. In addition, all categories in MIM-GAN are generated. The features (e.g., shape or angle) are different even though generated instances belong to the same category.

Here, we utilize the *MNIST Score* [42], which is similar to the *Inceptions Score* [43] but instead uses a classifier adapted to the MNIST training data instead of the Inception network to evaluate the quality of generated data. The *MNIST scores* (higher scores reflect better performance) are shown in Table 1. As is visually predictable from the outputs, the *MNIST score* of our proposed MIM-GAN is higher than other GAN variants.

Since our proposed MIM-GAN utilizes mutual information to guide different generators to capture different manifolds, and the

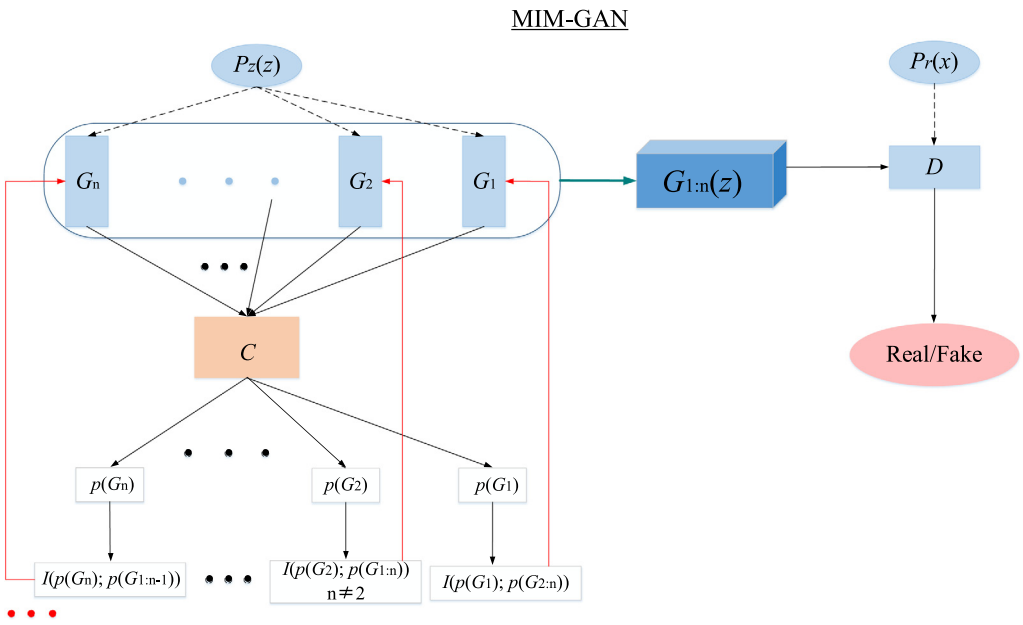


Fig. 3. Dotted arrows indicate sampling from a specific distribution. For reducing the correlation among generators, we employ mutual information to guide the generators mutually exclusively learn the manifolds of original data. The navy-blue arrow indicates that we combine all generated data to form a mixture generation $G_{1:n}(z)$. The corresponding distribution of $G_{1:n}(z)$ is expected to match the original data distribution p_r. $p(G_i)$ indicates the probability distribution of generated data which are produced by ith generator and accurately predicted by the classifier C. The red arrow indicates that we integrate the mutual information into the corresponding generator loss.

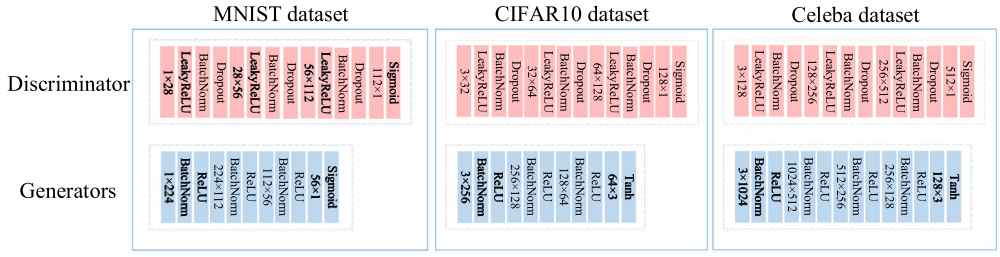


Fig. 4. Architectural details of MIM-GAN for MNIST and CIFAR10 as well as Celeba datasets. We use the *Adam* gradient method [37] and *BCE* [38] to update the parameters of all components. The parameter of LeakyRelu [32] is set to 0.02 and that of Dropout [39] is set to 0.5. The weights of our model are set to Normal(0.0, 0.02) and the biases are set to (0.0). The activation of the last layer for the discriminator and generator is Sigmoid [40] and Tanh [41], respectively. Note that the architectures of each generator are the same.

Table 1
The *MNIST scores* of generated data on MNIST dataset.

Models	*MNIST score*
WGAN [13]	6.006
MGAN [16]	6.618
SN-GAN [15]	6.514
DMGAN-PL [9]	6.622
RaGAN [36]	6.417
MIM-GAN	**7.958**

mutual information in this paper can reflect which manifolds the generators captured, we utilize this mutual information to observe the process of convergence of our model. In this case, we employ two generators and each generator produces more synthetic instances (here it is 50). The convergence process is shown in Fig. 6. From Fig. 6(c), a scenario can be obviously observed in which the mutual information converges at epoch=24. Fig. 6(a) and (b) shows the generated images (the first 5 rows are from one generator and the second 5 rows are from another generator) at epoch=24 and epoch=100. In sub-figure (a), the first generator has captured the categories '0', '1', '2', '4', '5' and '7' but lacks the figure '8', the second generator mainly holds the categories '3', '6', '8' and '9' but lacks the category '1'. The scenario of sub-figure (b) is similar to that of sub-figure (a) in that the first generator has captured the categories '3', '4', '6', '7' and '9' and the second generator holds the categories '0', '1', '2', '5' and '8'. However, the first generator and the second generator show all figures between the two cases combined. The scenario at epoch=200 is similar to both epoch=24 and epoch=100. This indicates that all categories can always be captured by our proposed MIM-GAN if the mutual information converges.

With each generator of MIM-GAN in Fig. 5 just producing 10 samples, one may wonder about the generating capability when producing more samples from each generator. To this end, we demonstrate each generator in MIM-GAN producing 64, 100 and 144 samples, respectively. Here, we employ two generators for convenient observation. The generated images are shown in Fig. 7. In sub-figure (a), the generated instances from G_2 do not hold the figure '2', but the generated instances from G_1 do not hold the figures '5' and '6'. In sub-figure (b), the same scenario (lacking figure '2') arises in G_2 while the generated instances from G_1 lack figures '6' and '8'. As for sub-figure (c), the generates instances from G_2 lack figure '4' while that from G_1 lacks figure '2'. However, G_1 and G_2 combined contain all figures. The experimental results show that generators in MIM-GAN learn

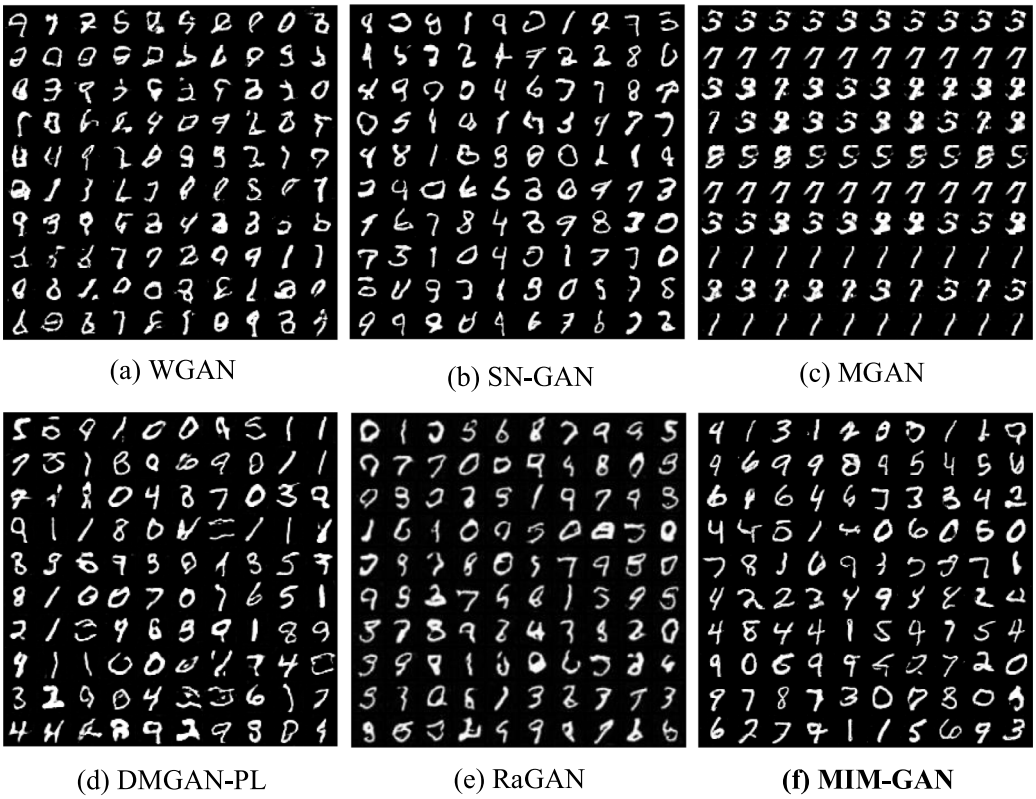


Fig. 5. Generated images by baselines and MIM-GAN. For MGAN, DMGAN-PL and MIM-GAN, we employ 10 generators and each generator only produces 10 samples. Each row indicates a generator.

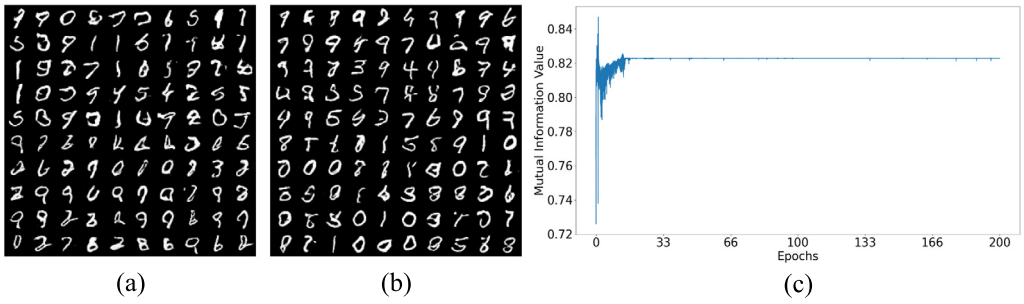


Fig. 6. The mutual information value of our proposed MIM-GAN on MNIST.

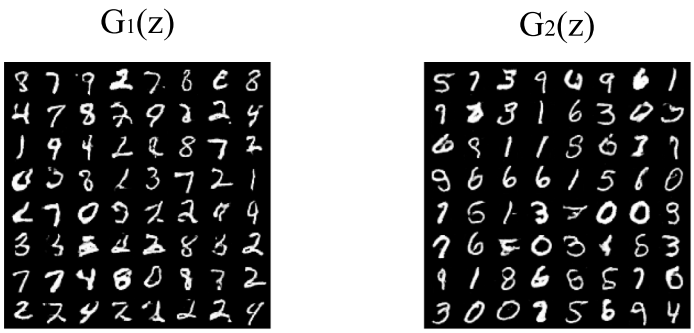
mutually exclusively from each other. Some manifolds learned by one generator are not learned by a different generator, thus addressing the problem of mode collapse.

5.2. CIFAR10 dataset

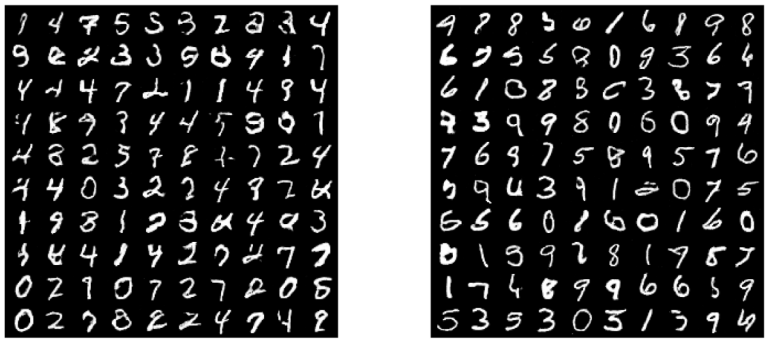
We continue to apply our proposed MIM-GAN and baselines to the CIFAR10 dataset. The 10 categories of CIFAR10 are 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship' and 'truck', respectively. In a similar manner to MNIST, we adopt 10 generators to produce simulation instances for MGAN, DMGAN-PL and MIM-GAN. The generated results are shown in Fig. 8, and the *Inception* scores and the *Fréchet Inception Distance* (FID) scores [44] obtained by our proposed MIM-GAN and baselines are reported in Table 2. Note that all objects in Fig. 8(b) generated by SN-GAN take the same color, and this can be viewed as a manifestation of mode collapse. This is because SN-GAN utilizes $\frac{W_{ij}}{\lambda_1}$ (W indicates the weight matrix of the discriminator and λ_1 refers to the largest singular value) to enforce the discriminator 1-Lipshitz continuity. However, such a strategy does not completely correspond to penalizing the spectral norm [19], and it limits, in turn, the diversity of generated data, resulting in all generated objects taking the same color. Under such a scenario, we cannot guarantee the generated data quality, which is loyally reflected by the *Inception* score 3.905. The color of generated images produced by MGAN either tends toward green or orange. The synthetic images produced by the DMGAN-PL model hold the worst quality, because the DMGAN-PL model prefers learning the prior over generators to produce synthetic images with good quality [9]. As with MNIST, we still utilize mutual information to observe the process of convergence of our model, which is shown in Fig. 9. From Fig. 9, we can observe that the mutual information gets converged at epoch=28. Overall, our proposed MIM-GAN outperforms the baselines, achieving the highest *Inception Score*. This result further proves that MIM-GAN helps address the problem of mode collapse.

5.3. Celeba dataset

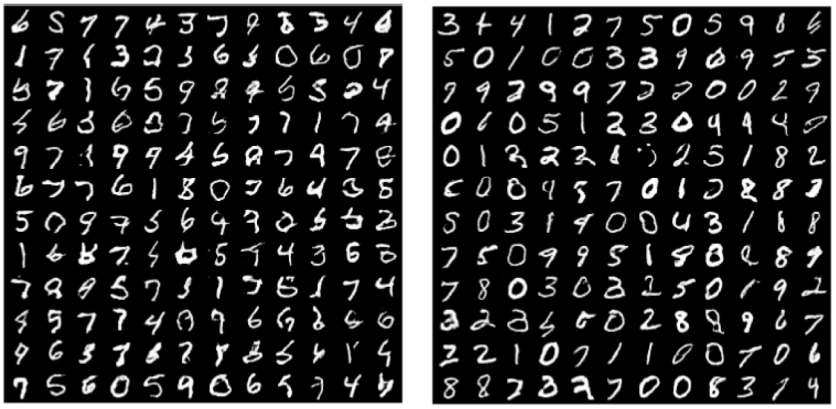
For conveniently training networks, we reshape the rectangle image to size $3 \times 128 \times 128$. In this case, we employ 5 generators



(a) Each generator generates 64 instances



(b) Each generator generates 100 instances



(c) Each generator generates 144 instances

Fig. 7. MIM-GAN employs two generators to produce simulation data, and each generator produces 64, 100 and 144 instances respectively. The left part indicates the first generator, and right part indicates the second generator.

Table 2
Inception scores and *FID scores* on CIFAR10 dataset.

Models	*Inception score*	*FID score*
WGAN [13]	4.035	218.195
MGAN [16]	4.392	226.907
SN-GAN [15]	3.905	229.135
DMGAN-PL [9]	2.758	223.826
RaGAN [36]	4.779	210.921
MIM-GAN	**6.233**	**203.954**

for MGAN, DMGAN-PL and our proposed MIM-GAN, and each generator only produces 5 instances. For calculating the mutual information, we manually label the faces with hats and glasses as 0 (and without hats as 1). The generated images are shown in Fig. 10. The MIM-GAN is, again, better than other GAN variants. The generated instances in sub-figure (a) are produced by WGAN, and the instances are all, basically, smiling faces. Furthermore, there are no people wearing hats or glasses. The generated images of SN-GAN (sub-figure (b)) and DMGAN-PL (sub-figure (d)) are covered by a gray or brown tint and have the lowest quality. For MGAN (sub-figure (c)), the performance is similar to WGAN. In other words, these generated faces have no hats or glasses. MIM-GAN produces a face with glasses (row 4, column 1 and column 5) and a face with a hat (row 5, column 5) along with intermixed smiling faces. This demonstrates that different generators learned different manifolds. The results show the effectiveness of MIM-GAN on the dataset at high resolution. Moreover, the convergence process of our model in CelebA is shown in Fig. 11, and the mutual information gets converged at epoch=72. In this case, we still employ two generators and each generator only produces 50 samples, which are shown in sub-figures (a) and (b). We now

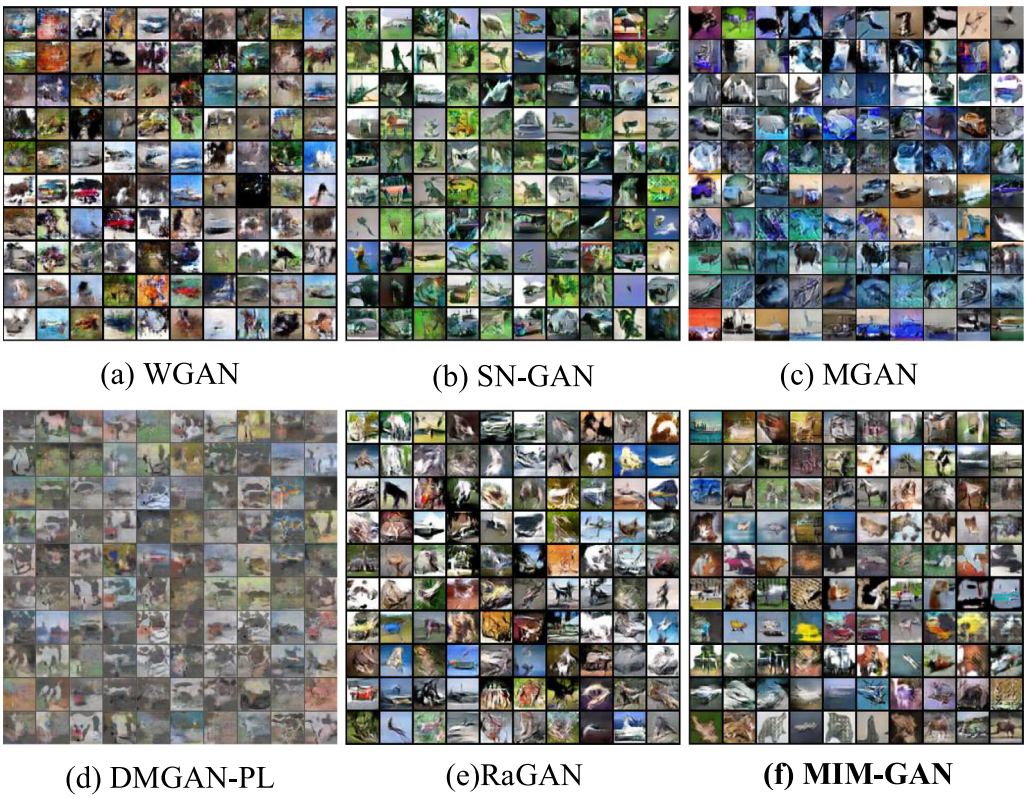


Fig. 8. Generated images by baselines and MIM-GAN. Similar to MNIST, each row indicates a generator in sub-figures (c), (d) and (f).

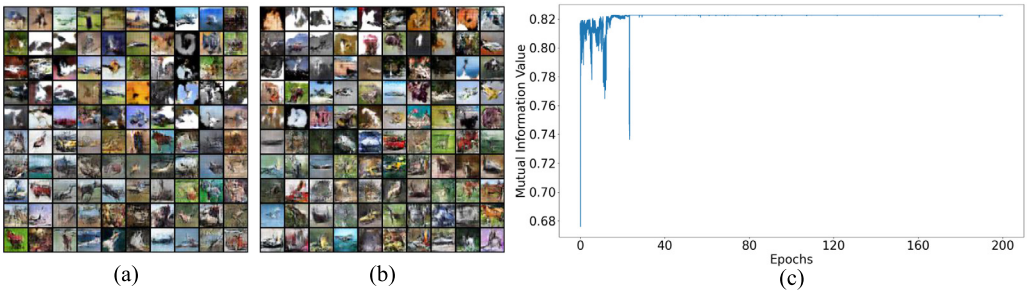


Fig. 9. The mutual information value of our proposed MIM-GAN on CIFAR10.

Table 3
FID scores of generated data on CelebA.

Models	FID score
WGAN [13]	250.724
MGAN [16]	248.674
SN-GAN [15]	231.094
DMGAN-PL [9]	377.280
RaGAN [36]	240.029
MIM-GAN	**186.528**

report the FID score obtained by our proposed MIM-GAN and the baselines in Table 3. From Table 3, we can observe that our proposed MIM-GAN outperforms baselines in terms of achieving the lowest FID score on CelebA dataset.

5.4. Discussion

Models utilizing multiple generators are a necessity in effectively learning the disconnected manifolds of original data [9]; as such, recent variants of GAN adopt this strategy to address the problem of mode collapse [9,16,23,24]. To guide multiple generators learning different manifolds, the current multi-generator GAN variants (e.g., MGAN and DMGAN-PL) focus on maximizing the correlation between generator id and generated samples [9, 16], and the models assume the generator ids either follow a certain distribution (e.g., Uniform or Multinomial) or utilize a neural network to approximate the distribution over generators. However, these strategies have challenges. The former tends to learn the same manifolds and produce identical instances, while the latter causes models to produce synthetic images with low quality. This study proposes to utilize the statistical probabilities of generated data to show which manifolds are captured by which generators. The correlation between generators is, additionally, minimized with mutual information to guide generators in learning different manifolds. This results in our model's generation of higher-quality and more diverse objects. See Fig. 5, Table 1, Fig. 8, Table 2 and Fig. 10. In addition, our proposed approach can significantly reduce the probability of having a negligible overlapping area between p_r and p_G. This means that MIM-GAN can also address the vanishing gradient problem.

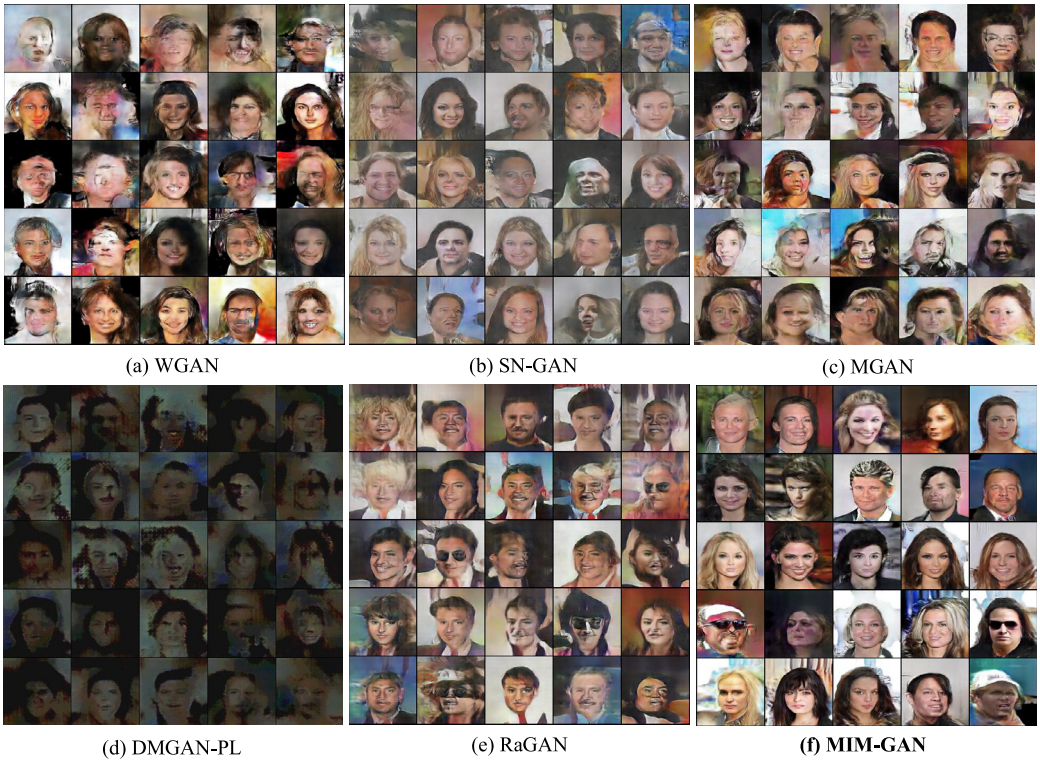


Fig. 10. Generated images by baselines and MIM-GAN. For MGAN, DMGAN-PL and MIM-GAN, we employ 5 generators and each generator only produces 5 samples. Each row indicates a generator.

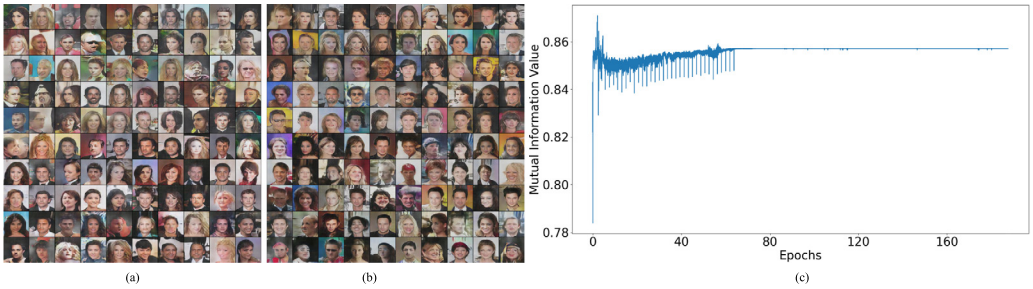


Fig. 11. The mutual information value of our proposed MIM-GAN on CelebA.

6. Conclusion

In this paper, to deal with the problem of mode collapse, we propose MIM-GAN. Our approach employs multiple generators, a classifier and a discriminator to learn a set of disconnected manifolds. Each generator is expected to capture the manifolds that the other generators do not hold. To achieve this goal, we adopt mutual information and propose a new minimax game among one discriminator, one classifier and a set of generators. We formulate an optimization problem that minimizes the mutual information among generators and minimizes the JS divergence between a mixture generated data distribution and an original data distribution. Comprehensive experiments on the MNIST, the CIFAR10 and the Celeba datasets demonstrate the following capabilities of our proposed MIM-GAN. (*i*), MIM-GAN generates diverse objects at different resolutions (28×28 for MNIST and 128×128 for Celeba). (*ii*), a set of disconnected manifolds are learned by multiple generators, and mutual information can prevent multiple generators from learning the same manifold. (*iii*), MIM-GAN achieves the highest *Inception score* and the lowest *FID score* among tested models.

Note that although the multi-generator GAN model is advantageous in addressing the problem of mode collapse, the theoretical analysis to determine the optimal number of generators is currently lacking in the multi-generator GAN community. However, we can observe some interesting empirical explanations in our paper. From Figs. 5 and 7, for example, we can easily observe that more generators are suitable for addressing the mode collapse problem if each generator produces few instances (e.g., 10 instances in Fig. 5), and few generators are enough if each generator produces more instances (e.g., 100 instances in Fig. 7). Based on the experimental results, we assume that the number of generators is relevant to the number of synthetic instances. We would like to investigate this hypothesis in our future work.

CRediT authorship contribution statement

Wei Li: Conceptualization, Investigation, Methodology, Writing - original draft. **Zhixuan Liang:** Data curation, Investigation, Software, Visualization. **Julian Neuman:** Project administration, Investigation, Writing - review & editing. **Jinlin Chen:** Conceptualization, Formal analysis, Investigation, Supervision, Validation. **Xiaohui Cui:** Resources, Funding acquisition, Supervision, Project administration, Writing - review & editing.

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

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

References

[1] 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.
[2] Wei Li, Linchuan Xu, Zhixuan Liang, Senzhang Wang, Jiannong Cao, Chao Ma, Xiaohui Cui, Sketch-then-edit generative adversarial network, Knowl.-Based Syst. (2020) 106102.
[3] Wei Li, Wei Ding, Rajani Sadasivam, Xiaohui Cui, Ping Chen, His-GAN: A histogram-based GAN model to improve data generation quality, Neural Netw. 119 (2019) 31–45.
[4] Zhangling Chen, Ce Wang, Huaming Wu, Kun Shang, Jun Wang, DMGAN: Discriminative metric-based generative adversarial networks, Knowl.-Based Syst. 192 (2020) 105370.
[5] Fei Wu, Xiao-Yuan Jing, Zhiyong Wu, Yimu Ji, Xiwei Dong, Xiaokai Luo, Qinghua Huang, Ruchuan Wang, Modality-specific and shared generative adversarial network for cross-modal retrieval, Pattern Recognit. (2020) 107335.
[6] Simon Kohl, David Bonekamp, Heinz-Peter Schlemmer, Kaneschka Yaqubi, Markus Hohenfellner, Boris Hadaschik, Jan-Philipp Radtke, Klaus Maier-Hein, Adversarial networks for the detection of aggressive prostate cancer, 2017, arXiv preprint arXiv:1702.08014.
[7] 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.
[8] Wei Li, Pengqiu Meng, Yi Hong, Xiaohui Cui, Using deep learning to preserve data confidentiality, Appl. Intell. 50 (2) (2020) 341–353.
[9] 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.
[10] A. Bordes, X. Glort, Y. Bengio, Deep sparse rectifier neural networks, Int. Conf. Artif. Intell. Stat. (2011) 315–323.
[11] Yang Guo, Dongsheng An, Xin Qi, Zhongxuan Luo, Shing-Tung Yau, Xianfeng Gu, et al., Mode collapse and regularity of optimal transportation maps, 2019, arXiv preprint arXiv:1902.02934.
[12] Min Lin, Qiang Chen, Shuicheng Yan, Network in network, 2013, arXiv preprint arXiv:1312.4400.
[13] Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein gan, 2017, arXiv preprint arXiv:1701.07875.
[14] 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. 5767–5777.
[15] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida, Spectral normalization for generative adversarial networks, 2018, arXiv preprint arXiv:1802.05957.
[16] Quan Hoang, Tu Dinh Nguyen, Trung Le, Dinh Phung, MGAN: Training generative adversarial nets with multiple generators, 2018.
[17] Alexander Kraskov, Harald Stögbauer, Peter Grassberger, Estimating mutual information, Phys. Rev. E 69 (6) (2004) 066138.
[18] Ludger Rüschendorf, The Wasserstein distance and approximation theorems, Probab. Theory Related Fields 70 (1) (1985) 117–129.
[19] Henry Gouk, Eibe Frank, Bernhard Pfahringer, Michael Cree, Regularisation of neural networks by enforcing lipschitz continuity, 2018, arXiv preprint arXiv:1804.04368.
[20] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. 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.
[21] Ning Zhang, Evan Shelhamer, Yang Gao, Trevor Darrell, Fine-grained pose prediction, normalization, and recognition, 2015, arXiv preprint arXiv:1511.07063.
[22] 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.
[23] 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-Volume 70, JMLR. org, 2017, pp. 224–232.
[24] 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.
[25] Constantinos Daskalakis, Paul W. Goldberg, Christos H. Papadimitriou, The Complexity of Computing a Nash Equilibrium, ACM, 2009.
[26] Sanjeev Arora, Andrej Risteski, Yi Zhang, Do GANs learn the distribution? some theory and empirics, 2018.
[27] Abhishek Kumar, Prasanna Sattigeri, Tom Fletcher, Semi-supervised learning with gans: Manifold invariance with improved inference, in: Advances in Neural Information Processing Systems, 2017, pp. 5534–5544.
[28] Solomon Kullback, Richard A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[29] Bent Fuglede, Flemming Topsoe, Jensen-Shannon divergence and Hilbert space embedding, in: International Symposium On Information Theory, 2004. ISIT 2004. Proceedings, IEEE, 2004, p. 31.
[30] Luke Metz Alec Radford, Soumith Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.
[31] Batch Normalization, Accelerating deep network training by reducing internal covariate shift, 2015, CoRR.–2015.–Vol. abs/1502.03167.–URL: http://arxiv.org/abs/1502.03167.
[32] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li, Empirical evaluation of rectified activations in convolutional network, Comput. Sci. (2015).
[33] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, Aleksander Madry, How does batch normalization help optimization?, in: Advances in Neural Information Processing Systems, 2018, pp. 2483–2493.
[34] Xiang Li, Shuo Chen, Xiaolin Hu, Jian Yang, Understanding the disharmony between dropout and batch normalization by variance shift, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2682–2690.
[35] Maurice Sion, et al., On general minimax theorems, Pacific J. Math. 8 (1) (1958) 171–176.
[36] Alexia Jolicoeur-Martineau, The relativistic discriminator: a key element missing from standard GAN, Int. Conf. Learn. Represent. (2019).
[37] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[38] 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.
[39] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[40] Jun Han, Claudio Moraga, The influence of the sigmoid function parameters on the speed of backpropagation learning, in: International Workshop on Artificial Neural Networks, Springer, 1995, pp. 195–201.
[41] Willy Malfliet, Willy Hereman, The tanh method: I. Exact solutions of nonlinear evolution and wave equations, Phys. Scr. 54 (6) (1996) 563.
[42] Corentin Hardy, Erwan Le Merrer, Bruno Sericola, Md-gan: Multi-discriminator generative adversarial networks for distributed datasets, 2018, arXiv preprint arXiv:1811.03850.
[43] Shane Barratt, Rishi Sharma, A note on the inception score, 2018, arXiv preprint arXiv:1801.01973.
[44] 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 Information Processing Systems, 2017, pp. 6626–6637.