Differential-Critic GAN: Generating What You Want by a Cue of Preferences

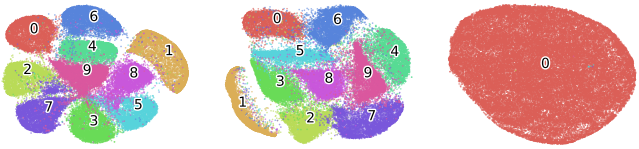
Yinghua Yao, Yuangang Pan✉, Ivor W. Tsang, *Fellow, IEEE,* and Xin Yao, *Fellow, IEEE*

***Abstract*—This paper proposes Differential-Critic Generative Adversarial Network (DiCGAN) to learn the distribution of user-desired data when only partial instead of the entire dataset possesses the desired property. DiCGAN generates desired data that meets the user's expectations and can assist in designing biological products with desired properties. Existing approaches select the desired samples first and train regular GANs on the selected samples to derive the user-desired data distribution. However, the selection of the desired data relies on global knowledge and supervision over the entire dataset. DiCGAN introduces a differential critic that learns from pairwise preferences, which are local knowledge and can be defined on a part of training data. The critic is built by defining an additional ranking loss over the Wasserstein GAN's critic. It endows the difference of critic values between each pair of samples with the user preference and guides the generation of the desired data instead of the whole data. For a more efficient solution to ensure data quality, we further reformulate DiCGAN as a constrained optimization problem, based on which we theoretically prove the convergence of our DiCGAN. Extensive experiments on a diverse set of datasets with various applications demonstrate that our DiCGAN achieves state-of-the-art performance in learning the user-desired data distributions, especially in the cases of insufficient desired data and limited supervision.**

***Index Terms*—Generative adversarial network, desired data generation, user preference, pairwise ranking.**

I. Introduction

LEARNING a good generative model for high-dimensional natural signals, such as images [1], video [2], speech [3] and text [4], has long been one of the key milestones of machine learning. Powered by the learning capabilities of deep neural networks, Generative Adversarial Networks (GANs) [5] have brought the field closer to attaining this goal. Currently, GANs are applied in a setting where the whole training dataset is of user interest (Fig. 1b). However, regular GANs no longer meet our requirement when only partial instead of the entire training dataset possesses the desired property [6]. Studying GANs under this setting can be useful in many real-world applications. One application can be to optimize generated biological data for desired properties, which can automate the process of designing DNA sequences, proteins and additional macromolecules for usage in medicine and manufacturing [7]. Another can be to generate images that meet the user's interest for image search [8].



(a) Training data (b) GAN (c) DiCGAN

Fig. 1: t-SNE of $50K$ MNIST samples from (a) training data, (b) GAN and (c) DiCGAN, respectively. Training on MNIST, DiCGAN learns the distribution of small digits, i.e., digit zero, while GAN learns the distribution of the entire dataset.

Existing methods [9], [10], [7] derive a user-desired data distribution by labeling the whole training dataset with a universal criterion. Based on the criterion, each training sample is annotated as "desired" or "undesired". Then GANs only learn to match the distribution of desired data. However, the requirement for the universal criterion is harsh since it requires global knowledge over the dataset, i.e., "what desired data is" derived from the whole data, which is expensive and is not available in many real-world applications [11], [7]. For example, when it is asked to train a robot to clean a table. It's not clear how to construct a suitable reward function (global knowledge, assessing all behaviors with a universal criterion). Secondly, labeling all of the training data causes high labor costs. Suppose that the user is interested in the generation of small digits on MNIST. The global knowledge is about the global ranking list of digits, i.e., $0 \succ 1 \ldots \succ 9$, where the notation $\succ$ denotes the left-hand side is preferred over the right-hand side. Traversing the whole dataset, the user annotates zero digits as "desired" and other digits as "undesired".

Instead of soliciting global knowledge, we consider an easier setting where GAN can be guided towards the distribution of user-desired data by user preferences. In particular, pairwise preferences are the most popular form of user preferences due to their simplicity and easy accessibility [12]. Such supervision only requires local knowledge, which can be easily collected with local ranking information. For training a robot to clean a table, it is feasible to compare two behaviors of the robot and determine which one is preferred w.r.t. the goal of cleaning the table [11]. In addition, resorting to pairwise preferences, it is not necessary to label all training data. Therefore, our target is

Yinghua Yao is with Guangdong Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, and also with Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. Yuangang Pan and Ivor W. Tsang are with A*STAR Center for Frontier AI Research, Singapore. Ivor W. Tsang is also with Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. Xin Yao is with the Research Institute of Trustworthy Autonomous Systems (RITAS) and Guangdong Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, and also with School of Computer Science, University of Birmingham, UK. Email: yinghua.yao@student.uts.edu.au, yuangang.pan@gmail.com, ivor.tsang@gmail.com, xiny@sustech.edu.cn. (Corresponding author: Yuangang Pan)

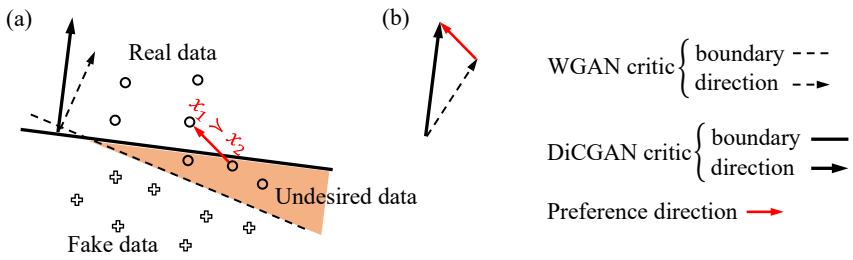


Fig. 2: Illustration of why DiCGAN can learn the user-desired data distribution. (a) DiCGAN's critic pushes fake data towards the real desired data while WGAN's critic pushes fake data towards all the real data. (b) The change of DiCGAN's critic direction is driven by the preference direction. Note that the preference direction is learned from all pairwise preferences.

to incorporate pairwise preferences into the learning of GAN, so as to guide the generation of the desired data.

Relativistic GAN (RGAN) [13] is a variant of regular GAN and is proposed to learn the whole data distribution. It considers the critic values as the indicators of sample quality, which are similar to the ranking scores. Motivated by this, we consider taking the critic values as the ranking scores and define the ranking loss for pairwise preferences based on the critic values directly. In particular, the difference in critic values for each pair of samples reflects the user's preference over the samples. This is why we call our critic the **differential critic**, and we propose Differential-Critic GAN (DiCGAN) for learning the user-desired data distribution. As shown in Fig. 2, the differential critic incorporates the direction of user preferences, which pushes the original critic direction towards the real desired data region instead of the entire real data region.

The main contributions are summarized as follows:

- We propose DiCGAN to learn the distribution of the desired data from the entire dataset (Fig. 1c) using pairwise preferences. To the best of our knowledge, this is the first work to use local knowledge, i.e., local ranking information about user preferences to learn the desired data distribution. Such knowledge is wildly accessible and can be defined on part of the training data.
- We are the first one to endow the difference in the critic values between each pair of samples with user preferences. In particular, we incorporate user preferences into GAN's learning to build Differential-Critic GAN (DiCGAN) via introducing an additional pairwise ranking loss over the WGAN's critic. Further, we propose an equivalent form of DiCGAN with a hard constraint to ensure data quality.
- We theoretically prove that the distribution of generated samples in DiCGAN can converge to the distribution of desired samples. The relationship between the user preferences and the distribution distance is the first time to be rigorously shown.
- We empirically study that our DiCGAN can generate images that meet the user's interest on MNIST and CelebA-HQ and help design biological products with desired properties on the gene sequence dataset. Our DiCGAN outperforms the baselines in learning the distribution of desired data especially when labels of desired data are limited.

II. Background

A. Related Work

Vanilla GANs, like original GAN [5], Wasserstein GAN (WGAN) [9], can be adapted to learn a user-desired data distribution. A naive way is to first select the samples possessing the desired property based on a universal criterion and then perform regular GAN training only on the selected samples to derive the desired data distribution. However, the criterion needs to give a global ranking over the whole data in terms of the interested property so as to pick up desired data, which is expensive. It may not be accessible in real applications. Even, these GANs will fail when the desired samples are insufficient.

The conditional variants of GAN [10], [14] can be applied in this setting by modeling "desired/undressed" labels as condition variables to learn the conditional desired data distribution. However, the splitting of desired data and undesired data also requires a universal criterion. On the other hand, the generation performance of condition-based GAN is governed by the respective conditions with sufficient training observations. When the desired data is limited, the conditional modeling is dominated by the major classes, i.e., undesired data, resulting in a failure to capture the desired data distribution.

Feedback GAN (FBGAN) [7] successfully derives the user-desired data distribution with limited desired data by iteratively introducing desired samples into the training data. Specifically, FBGAN is pre-trained with all training data using the vanilla GAN. At each training epoch, the generator first generates certain amounts of samples. The generated samples possessing the desired property are selected by a universal criterion and used to replace the old training data. Then, regular GAN is trained with the updated training data. Since the ratio of the desired samples gradually increases in the training data, all training data will be replaced with the desired samples. Finally, FBGAN would obtain the desired data distribution. Instead of explicitly selecting desired samples, an intuitive way is to first pre-train GAN on all data and then fine-tune it with a classification loss of classifying the generation as "desired" to derive the desired data distribution. However, these methods are still restrictive due to the requirement of the universal criterion.

All literature methods resort to a universal criterion to select the desired data in order to learn the desired data distribution, but the criterion requires expensive global knowledge and may even not exist in real applications. In addition, all methods need to label the entire training data, which incurs huge costs. Our work derives the user-desired data distribution using pairwise preferences, which only requires local knowledge and can reduce the burden of labeling the whole training data [15], [16], [17]. Therefore, our DiCGAN has the advantage of requiring less and more accessible supervision than existing approaches.

B. Preliminaries: Generative Adversarial Networks

GAN [5] performs generative modeling by learning a map from low-dimensional latent space $\mathcal{Z}$ to data space $\mathcal{X}$, i.e.,

$G : \mathcal{Z} \to \mathcal{X}$, given samples from the training data distribution, namely, $x \sim p_{\mathrm{r}}(x)$. The goal is to find G that achieves $p_{\theta}(x) = p_{\mathrm{r}}(x)$, where $p_{\theta}(x)$ is the distribution of fake data $x = G(z)$.

In order to train the generator, GAN introduces another network, i.e., discriminator, to discriminate real data from fake data. The generator is trained to produce images that are conceived to be realistic by the discriminator. Two networks are trained alternately until the generator successfully fools the discriminator. GAN [5]'s objective is defined as follows:

$$\min_{G} \max_{D} \mathbb{E}_{p_{\mathrm{r}}(x)} \left[\log \sigma \left(D(x)\right)\right] + \mathbb{E}_{p_{\theta}(x)} \left[\log \left(1 - \sigma \left(D(x)\right)\right)\right], \tag{1}$$

where $\sigma \left(D(x)\right)$ is the probability that the input data is real and σ is the sigmoid function. $D(x)$ is the non-transformed discriminator output, which is called *critic value* in WGAN [9].

WGAN [9], [18] and RGAN [19] are stable variants of GANs defining the loss functions in terms of the critic D, i.e., the non-transformed discriminator. Specifically, WGAN measures the quality of fake data in terms of the Wasserstein distance (W-distance) between the real data distribution and the fake data distribution. The W-distance is approximated by the difference in the average critic values between the real data and the fake data. WGAN's objective is defined as follows:

$$\min_{G} \max_{D} \mathbb{E}_{p_{\mathrm{r}}(x)} \left[D(x)\right] - \mathbb{E}_{p_{\theta}(x)} \left[D\left(x\right)\right], \tag{2}$$

where D is the critic enforced with a 1-Lipschitz constraint.

RGAN estimates the probability that the given real data is more realistic than randomly sampled fake data by using the difference in the critic values. Its objective is defined as follows:

$$\begin{aligned} &\max_{D} \mathbb{E}_{x_{\mathrm{r}} \sim p_{\mathrm{r}}(x), x_{\theta} \sim p_{\theta}(x)} \left[\log \left(\sigma \left(D(x_{\mathrm{r}}) - D\left(x_{\theta}\right)\right)\right)\right], \\ &\max_{G} \mathbb{E}_{x_{\mathrm{r}} \sim p_{\mathrm{r}}(x), x_{\theta} \sim p_{\theta}(x)} \left[\log \left(\sigma \left(D(x_{\theta}) - D\left(x_{\mathrm{r}}\right)\right)\right)\right]. \end{aligned} \tag{3}$$

It has a similar form as the pairwise ranking loss [20], but interpreting the critic values as ranking scores for sample quality.

Our DiCGAN considers the critic values as the ranking scores, which has a similar viewpoint to RGAN. But very differently, 1) our DiCGAN aims to learn the distribution of user-desired data when only part of the dataset possesses the desired property while RGAN targets to learn the whole data distribution; 2) our DiCGAN uses critic values to represent user preferences while RGAN uses critic values to describe data quality. We summarize the comparison of our DiCGAN with WGAN and RGAN in Table I. Our DiCGAN can be applied to GAN variants based on the critic, like WGAN and RGAN. In this work, we develop our DiCGAN on WGAN.

TABLE I: Comparison of DiCGAN with WGAN and RGAN in terms of the target data distribution and the critic value.

Method	target distribution	critic value	
		scope	physical meaning
WGAN	whole	distribution level	distribution distance
RGAN	whole	sample level	data quality
DiCGAN	partial (desired)	sample level	user preference

III. DiCGAN for User-desired Distribution

No longer learning the distribution of the whole dataset, GAN is applied in a new scenario, where the distribution of the partial dataset is what we desire. User-desired data may refer to some certain class of data among multiple class datasets, or observations with/without some particular attributes or properties. Such data can be induced from user preference, which can be represented as an ordering relation between two or more samples in terms of the desired property. We propose differential-critic GAN (DiCGAN) to learn the desired data distribution from the user preferences along with the whole dataset.

TABLE II: Main mathematical notations in this paper.

Notation	Explanation
G	generator
D	critic
X	training samples
$x_1 \succ x_2$	x_1 is preferred over x_2
S	pairwise preferences, $\mathrm{S} = \left\{s = (x_1, x_2) \middle| x_1 \succ x_2, x_1, x_2 \in \mathrm{X}\right\}$
$p_{\mathrm{r}}(x)$	the distribution of the whole data
$p_{\mathrm{d}}(x)$	the distribution of the user-desired data
$p_{\mathrm{u}}(x)$	the distribution of the undesired data
$p_{\theta}(x)$	the target generative model
T	threshold to discriminate the desired data from the undesired data
$f(x)$	score function that maps sample x to the score that reflects the user's preference for x
$d(,)$	distribution distance
ε	distance constraint that guarantees good generation quality

A. Learning the Distribution of User-desired Data

A universal criterion to help derive a user-desired data distribution can be constructed based on a score function. Following the score-based ranking literature [21], we suppose that there exists a numeric score associated with each sample, reflecting the user's preference for the sample. A higher score indicates that its corresponding sample is preferred by the user. In detail, let $f()$ denote a score function that maps sample x to score $f(x)$. Let T denote the threshold to discriminate the desired data from the undesired data. That is, if a sample's score $f(x)$ exceeds a predefined threshold T, namely, $I(f(x) > T) = 1$, the sample x is desired by the user. $I()$ is a sign function, which equals 1 if its condition is true and 0 otherwise. For the sake of explanation, we use $p_{\mathrm{r}}(x), p_{\mathrm{d}}(x), p_{\mathrm{u}}(x)$ to denote the distribution of the whole data, the user-desired data and the undesired data, respectively.

Current literatures [7], [9], [10] needs to explicitly label desired/undesired data in order to learn the distribution of the desired data $p_{\mathrm{d}}(x)$. Namely, the desired data $\mathrm{X_d} = \{x | I(f(x) > T) = 1, x \sim p_{\mathrm{r}}(x)\}$. The undesired data $\mathrm{X_u} = \{x | I(f(x) \leq T) = 1, x \sim p_{\mathrm{r}}(x)\}$. However, the assumption that the score function $f()$ is predefined may be too restrictive for real applications, where no universal and explicit criteria exist. Second, the definitions of the desired/undesired samples are highly dependent on the choice of the threshold T. Third, labeling over the entire dataset incurs high costs.

Instead of relying on a predefined score function (global knowledge), we propose to learn the desired data distribution in a straightforward manner from the user preferences. Here, we consider general auxiliary information, i.e., the pairwise preferences, to represent the user preferences, due to its simplicity and easy accessibility. For any two samples $x_1, x_2 \sim p_{\mathrm{r}}(x)$, let $x_1 \succ x_2$ denote that x_1 is preferred over x_2 according to the user's preference over the samples. Let X be the training

samples, i.e., $\mathrm{X} = \{x \sim p_{\mathrm{r}}(x)\}$. A collection of pairwise preferences S is obtained by:

$$\mathrm{S} = \left\{ s = (x_1, x_2) | x_1 \succ x_2, x_1, x_2 \in \mathrm{X} \right\}. \tag{4}$$

S can be defined on part of the dataset.

Remark 1. *We can construct* S *by first randomly drawing sample pairs from part of the training samples and then asking the user to select the preferred one from each pair.*

Definition 1 (Problem Setting)**.** *Given the training samples* X *and the pairwise preferences* S*, the target is to learn a generative model $p_\theta(x)$ that is identical to the distribution of the desired data $p_{\mathrm{d}}(x)$, i.e., $p_\theta(x) = p_{\mathrm{d}}(x)$.*

B. Differential Critic GAN

Instead of adopting WGAN's critic for quality assessment, we present the differential critic for modeling pairwise preferences. The differential critic can guide the generation of the user-desired data.

1) Pairwise Preference: We consider incorporating pairwise preferences into the training of GAN.

The score-based ranking model [22] is used to model the pairwise preferences. It learns the score function $f()$, of which the score value, called ranking score in the model, is the indicator of the user preferences. Further, the difference in ranking scores can indicate the pairwise preference relation. That is, for any pair of samples x_1, x_2, if $x_1 \succ x_2$ then $f(x_1) - f(x_2) > 0$ and vice versa. For any pairwise preference $s : x_1 \succ x_2$, the ranking loss we consider is as follows:

$$h(s) = \max\left(0, -\left(f\left(x_1\right) - f\left(x_2\right)\right) + m\right), \tag{5}$$

where m is the ranking margin. For other forms of ranking losses, the reader can refer to [22].

Instead of learning the score function independently of GAN's training, we consider incorporating it into GAN's training, guiding GAN towards the generation of the desired data. The critic in RGAN [13] is similar to the score function, where the critic values are used to describe the quality of samples. We are motivated to take the critic values as the ranking scores and define the ranking loss on the critic directly. In particular, the difference in the critic values for each pair of samples reflects the user's preference over the samples.

2) Loss Function: We build DiCGAN based on WGAN and the pairwise ranking loss is defined over the WGAN's critic. The loss function for DiCGAN is defined as:

$$\min_G \max_D \mathbb{E}_{p_{\mathrm{r}}(x)}\left[D(x)\right] - \mathbb{E}_{p_\theta(x)}\left[D\left(x\right)\right] - \lambda \frac{1}{|\mathrm{S}|} \sum_{s \in \mathrm{S}} \left[h\left(s\right)\right], \tag{6}$$

where $h(s)$ is the pairwise ranking loss (Eq. (5)). $f()$ is approximated by the critic D. Namely, $h(s) \approx \max\left(0, -\left(D\left(x_1\right) - D\left(x_2\right)\right) + m\right)$. λ is a balance factor, which will be discussed further in section III-C. Similar to WGAN, we formulate the objective for the differential critic L_D and the generator L_G as:

$$\begin{aligned} L_D &= \frac{1}{b} \sum_{i=1}^{b} \left(D(x^i) - D(G(z^i))\right) - \lambda \frac{1}{n_{\mathrm{s}}} \sum_{j=1}^{n_{\mathrm{s}}} h(s^j), \\ L_G &= \frac{1}{b} \sum_{i=1}^{b} -D(G(z^i)). \end{aligned} \tag{7}$$

where b is the batch size. n_{s} is the number of preferences sampling from S.

The advantages of DiCGAN are twofold. (1) The introduced ranking loss in DiCGAN is defined on the critic directly. Apart from WGAN, it can be easily applied to other GAN variants developed based on the critic, e.g., RGAN. (2) The construction of pairwise preferences involves the undesired data. Thus, the undesired samples are also utilized during the training and they, together with desired samples, provide the generation direction of the desired data for the generator.

We argue that the differential critic in DiCGAN can guide the generator to learn the user-desired data distribution. As shown in Fig. 2, the differential critic in DiCGAN provides the direction towards the real desired data. We denote the critic direction as the moving direction of the fake data, which is orthogonal to the decision boundary of the critic. Referring to Eq. (6), DiCGAN's critic loss consists of two terms: the vanilla WGAN loss and the ranking loss. The vanilla WGAN loss imposes the critic direction from the fake data to the real data. Meanwhile, the ranking loss induces a user preference direction, which points from the undesired data to the desired data. Combining these two effects, the critic direction of DiCGAN targets the region of the real desired data only.

The above proposed DiCGAN (Eq. (6)) however requires sensitive hyperparameter tuning during the training. Revisiting the objective (Eq. (6)), the first two terms (WGAN loss) can be considered as the WGAN regularization, which ensures the generated data distribution is close to the whole real data distribution, i.e., $p_\theta \approx p_{\mathrm{r}}$. The third term (ranking loss) serves as a correction for WGAN, which makes WGAN slightly biased to our target of learning the desired data distribution, i.e., $p_\theta = p_{\mathrm{d}}$. Therefore, the WGAN regularization serves as the cornerstone of our DiCGAN. Particularly, if the desired data distribution is close to the whole data distribution, the ranking loss easily corrects the WGAN to achieve the desired data distribution. Otherwise, satisfactory performance of DiCGAN may require the online hyperparameter tuning of λ during the training process. Thus, it is hard to train with Eq. (6) in this case.

C. Reformulating DiCGAN to Ensure Data Quality

In this section, we reformulate DiCGAN as a form with a hard constraint. This form indicates that the tuning for Eq. (6) relies largely on the distance between the distributions of the desired data and the undesired data. Further, it inspires us to derive a more efficient solution – minor correction and major correction.

According to the above analysis, the WGAN loss serves as the cornerstone of our DiCGAN and the pairwise ranking loss serves as a correction for WGAN. Thus, we consider

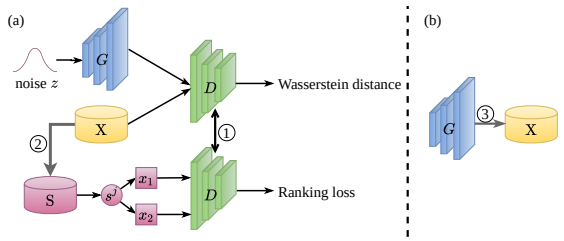


Fig. 3: DiCGAN architecture and training. DiCGAN is alternately trained with step (a) and (b). (a) Training DiCGAN at one minor correction. (b) Replacing data after one minor correction. ① denotes the shared differential critic D. ② denotes that S is constructed from X using Eq. (4). ③ denotes data replacement using Eq. (10).

reformulating the objective of DiCGAN, i.e., Eq. (6) into an equivalent objective with a hard WGAN constraint:

$$\begin{aligned} &\min_{G}\max_{D} -\sum_{s\in \mathrm{S}}\left[h\left(s\right)\right], \\ &\text{s.t. } d(p_{\mathrm{r}},p_{\theta}) = \left|\mathbb{E}_{p_{\mathrm{r}}(x)}\left[D(x)\right] - \mathbb{E}_{p_{\theta}(x)}\left[D\left(x\right)\right]\right| \leq \varepsilon. \end{aligned} \tag{8}$$

where $\varepsilon > 0$. Note that we impose an explicit non-negative constraint on $d(p_{\mathrm{r}}, p_{\theta})$, to highlight that it is a distance metric. It is still equivalent to WGAN loss from its definition. Eq. (6) is the Lagrangian function. Since Eq. (8) imposes a hard constraint on the WGAN loss, it is more difficult to optimize compared to Eq. (6). However, more efficient solutions of DiCGAN can be explored by analyzing Eq. (8) regarding the hard constraint on $d(p_{\mathrm{r}}, p_{\theta})$.

In terms of a minor correction situation, this means the desired data distribution p_{d} is close to the real data distribution p_{r}. Therefore, the hard constraint dominates the training goal of DiCGAN. By assigning a proper λ to ensure the constraint is satisfied, Eq. (6) can learn the distribution of the user-desired data while ensuring data quality.

In terms of a major correction situation, this means the desired data distribution p_{d} is quite diverse from the real data distribution p_{r}. Therefore, DiCGAN needs to achieve an equilibrium between the correction, imposed by the ranking loss, and the hard constraint, imposed by the WGAN loss. However, a large correction may not ensure the quality of the generated data, since the WGAN loss, used to guarantee the image quality, is defined between the generated data and the whole real data. To avoid the major correction, we propose to break the major correction into a sequence of minor corrections to ensure data quality. Namely, at each minor correction, we first use the generator G to generate n_{g} samples, denoted as $\mathrm{X_g}$:

$$\mathrm{X_g^e} \leftarrow \{G^e(z^1), \ldots, G^e(z^{n_{\mathrm{g}}})\}, \quad \{z^i \sim p(z)\}_{i=1}^{n_{\mathrm{g}}}, \tag{9}$$

where e is e-th minor correction. Then we replace partial old training samples with the generated samples:

$$\mathrm{X}^{e+1} \leftarrow \mathrm{X}^{e} \setminus \mathrm{X_o^e} \cup \mathrm{X_g^e}, \tag{10}$$

where $\mathrm{X_o^e}$ are the old (least-recently added) n_{g} samples in X^e.

Due to the ranking loss, the generated data distribution p_{θ}^e is closer to the desired data distribution p_{d}, compared to the constructed p_{r}^e at each minor correction. Therefore, the iterative replacement (Eq. (10)) can gradually shift the real data distribution p_{r} towards the desired data distribution p_{d}. Namely, $d(p_{\mathrm{r}}, p_{\mathrm{d}}) > \cdots > d(p_{\mathrm{r}}^e, p_{\mathrm{d}}) > d(p_{\mathrm{r}}^{e+1}, p_{\mathrm{d}}) > \cdots$. According to the monotone convergence theorem, $d(p_{\mathrm{r}}^e, p_{\mathrm{d}})$ will converge to zero when $e \to +\infty$. So only a minor correction needs to be imposed on p_{θ}^e by optimizing Eq. (6) at each minor correction. Iteratively, the generated distribution p_{θ} shifts towards p_{d}. The training algorithm is summarized in Algorithm 1. The architecture and training of DiCGAN can be seen in Fig. 3. For the sake of easy optimization, we pretrain the differential critic D and the generator G using vanilla WGAN.

Algorithm 1 Training algorithm of DiCGAN

Input: training data X, pairwise preferences S
Initilization: balance factor λ, #generated samples n_{g}, #pairs n_{s}, batch size b, #iterations per minor correction n_{i}, #critic iterations per generator iteration n_{critic}
Pretrain D and G
repeat
  % Shift to the user-preferred distribution
    Generate samples using Eq. (9)
    Replace partial old samples in X with $\mathrm{X_g}$ using Eq. (10)
  Obtain pairwise preferences S using Eq. (4)
  % Training of D and G at a minor correction
  for $l = 1, \ldots, n_{\mathrm{i}}$ **do**
    for $t = 1, \ldots, n_{\mathrm{critic}}$ **do**
      Sample $\{x^i\}_{i=1}^{b}$ from X, $\{z^i \sim p(z)\}_{i=1}^{b}$
      Sample $\{s^j\}_{j=1}^{n_{\mathrm{s}}}$ from S.
      Train the differential critic D using L_D in Eq. (7)
    end for
    Train the generator G using L_G in Eq. (7)
  end for
until converge
Output: generator G for desired data distribution

D. Convergence Analysis

In this section, we analyze the convergence of our DiCGAN under the minor correction and the major correction, respectively. In the case of the minor correction, we prove that the distribution of generated data $p_{\theta}(x)$ converges to the distribution of user-desired data $p_{\mathrm{d}}(x)$ via adversarial training of GAN given that the differential critic of DiCGAN converges to the score function whose score describes the user's preference for the sample. In the case of the major correction, $p_{\theta}(x)$ is proven gradually moving towards $p_{\mathrm{d}}(x)$ with a sequence of minor correction as one minor correction shifts $p_{\theta}(x)$ towards $p_{\mathrm{d}}(x)$ with a certain small distance.

Suppose the training data $\mathrm{X} = \{x_1, x_2, \ldots, x_n\}$, where n is the number of training samples. Their corresponding scores $o = f(x)$ are $\{o_1, o_2, \ldots, o_n\}$, which describes the user's preference for the sample. The maximum score among the samples is denoted as o_{max} while the minimum one is o_{min}.

Proposition 1. *In the case of the minor correction, i.e.,* $d\left(p_{\mathrm{r}}(x), p_{\mathrm{d}}(x)\right) \leq \varepsilon$, $p_{\theta}(x)$ *converges to* $p_{\mathrm{d}}(x)$.

Proof. According to the theory of learning to rank [17], by setting an appropriate λ, we have D converge to the score function $f()$ iff S is sufficient. Then the real desired data will be assigned higher scores than the real undesired data.

With Eq. (7), the generator is optimized to generate samples with scores as high as possible while the critic is optimized to assign the generated samples lower than the training samples. As only training samples preferred by the user are assigned with high scores, when the adversarial training converges, the generated samples are alike samples with high scores, i.e., the desired data, which shares the same principle in [5], [9]. Therefore, $p_\theta(x)$ converges to $p_d(x)$. □

Proposition 2. *In the case of the minor correction where* $d(p_r(x), p_d(x)) \leq \varepsilon$ *and* $p_\theta(x) = p_d(x)$*, we can prove that* $\mathbb{E}_{p_\theta(x)}[D(x)] = \mathbb{E}_{p_r(x)}[D(x)] + \delta$*, for some* $\delta > 0$*.*

Proof. Without loss of generality, we represent $p_r(x)$ and $p_\theta(x)$ in a fine-grain formulation. Namely,

$$p_r(x) = (1-\alpha)p_d(x) + \alpha p_u(x), \quad p_\theta(x) = p_d(x), \quad (11)$$

where $\alpha \in [0,1]$ is a very small value such as $d(p_r(x), p_d(x)) \leq \varepsilon$ is satisfied.

Furthermore, according to our definition of the ranking model, the score for the desired data should be higher than that of the undesired data, namely

$$D(x) \begin{cases} > T, & \text{if } x \sim p_d(x); \\ \leq T, & \text{if } x \sim p_u(x), \end{cases} \quad (12)$$

where $o_{min} < T < o_{max}$. (1) (o_{min}, o_{max}) is introduced since the critic score is always bounded; (2) T denotes some value to discriminate the desired data from undesired data.

Taking into consideration of both Eq. (11), (12), we have

$$\begin{aligned} \mathbb{E}_{p_r(x)}[D(x)] &= \int [(1-\alpha)p_d(x) + \alpha p_u(x)] D(x)dx \quad (13) \\ &= \int [(1-\alpha)p_d(x)] D(x)dx + \int [\alpha p_u(x)] D(x)dx \\ &= \int p_d(x)D(x)dx - \alpha(\int p_d(x)D(x)dx - \int p_u(x)D(x)dx). \end{aligned}$$

Considering that (1) $p_d(x)$ and $p_u(x)$ are always positive; (2) $D(x)$ is continuous and bounded on the domain of x with respect to $p_d(x)$ and $p_u(x)$, respectively, we have the following derivations according to the mean value theorem for integrals:

$$\exists\, \xi_d \in (T, o_{max}], \quad \int p_d(x)D(x)dx = \xi_d \int p_d(x)dx = \xi_d;$$
$$\exists\, \xi_u \in [o_{min}, T], \quad \int p_u(x)D(x)dx = \xi_u \int p_u(x)dx = \xi_u.$$

Since $\xi_d > \xi_u$, we have

$$\alpha\left(\int p_d(x)D(x)dx - \int p_u(x)D(x)dx\right) = \alpha(\xi_d - \xi_u) > 0.$$

$\Longrightarrow$ Eq. (13) $= \mathbb{E}_{p_d(x)}[D(x)] - \delta$, where $\delta = \alpha(\xi_d - \xi_u) > 0$. Furthermore, by replacing $p_d(x)$ with $p_\theta(x)$, we have

$$\mathbb{E}_{p_\theta(x)}[D(x)] = \mathbb{E}_{p_r(x)}[D(x)] + \delta,$$

for some $\delta > 0$. □

Corollary 2.1. *The minor correction moves* p_θ *towards* p_d *with distance* δ *compared to* p_r*, i.e.,* $d(p_r, p_d) - d(p_\theta, p_d) = \delta$*.*

Proposition 3. *In the case of the major correction, i.e.,* $d(p_r, p_d) = T_0$*, the distance between* $p_\theta(x)$ *and* $p_d(x)$ *converges to* $d(p_\theta^k, p_d) = T_0 - k\delta$ *after* k *minor corrections.*

Proof. The major correction is divided into a sequence of minor corrections. In the first minor correction, the training data distribution is $p_r(x)$. Derived by Corollary 2.1, after this minor correction, we have

$$d(p_r, p_d) - d(p_\theta^1, p_d) = \delta.$$

With $d(p_r, p_d) = T_0$, we can get

$$d(p_\theta^1, p_d) = T_0 - \delta.$$

In the second minor correction, we can replace all training samples with the generated samples obtained in the first correction. Thus, the training data distribution becomes $p_\theta^1(x)$. After two minor corrections, similarly, we can get

$$d(p_\theta^2, p_d) = T_0 - 2\delta.$$

So on and so forth. After k minor corrections, we can get

$$d(p_\theta^k, p_d) = T_0 - k\delta. \quad \square$$

Corollary 3.1. *In the case of major correction,* $p_\theta(x)$ *converges to* $p_d(x)$ *after* K *minor corrections, where* $K = \lceil \frac{T_0}{\delta} \rceil$*.*

Proof. Since $d(p_\theta^k, p_d) >= 0$ and $d(p_\theta^k, p_d) = T_0 - k\delta$ decreases as k increases, $d(p_\theta^k, p_d)$ converges to zero when $k \to +\infty$ according to the monotone convergence theorem. Specifically, when $k = \lceil \frac{T_0}{\delta} \rceil$, $d(p_\theta^k, p_d) = 0$. □

From Proposition 1 and Corollary 3.1, we conclude that in DiCGAN, the distribution of generated samples $p_\theta(x)$ converges to $p_d(x)$.

E. Technical Novelty of DiCGAN

We elaborate our DiCGAN's technical novelty and its significance in terms of the following four aspects:

- **The first one to apply user preferences for desired data generation.** Current approaches for desired data generation require expensive global knowledge, which is usually not available. Our DiCGAN uses local knowledge only – local ranking information about user preferences.
- **New insight for critic value.** Our DiCGAN considers the critic values as the ranking scores that represent user preferences. Based on this insight, we can incorporate user preferences into GAN's learning instead of learning the score function for user preferences independently of GAN's training:
  1. *Naive combination between user preferences and GAN does not work.* Introducing an additional critic that learns from user preferences onto WGAN would lead to the conflict between WGAN's original critic for good quality generation and the extra critic for desired data generation.
  2. As critic values can represent data quality and user preferences, we define a differential critic by defining an additional pairwise ranking loss on the WGAN's critic and build DiCGAN (Eq. (6)). Then the original WGAN's critic loss encourages:

$$x_1 > x_2 \text{ for } x_1 \sim p_d(x) \text{ and } x_2 \sim p_u(x);$$

and the ranking loss encourages:

$$x_1 > x_2 \text{ for } x_1 \sim p_{\mathrm{r}}(x) \text{ and } x_2 \sim p_{\theta}(x).$$

The critic would guide the generation with high critic values, encouraging the generation of user-desired data with good quality.

- **Efficient solution by an equivalent form with a hard constraint.** The naive form of DiCGAN (Eq. (6)) requires heavy hyper-parameter tuning when there is a large distance gap between the distributions of the desired data and the whole data. Thus, we propose an equivalent form of DiCGAN (Eq. (8)). Based on it, we derive a more efficient solution in terms of minor correction and major correction, which can always ensure good data quality.
- **The first rigorous model for desired data generation.** To the best of our knowledge, no previous work theoretically studies this problem. The above three points pave the way for the theoretical convergence proof of desired data generation:
  1. Because of DiCGAN's form with a hard distance constraint, we can analyze the convergence of DiCGAN under the minor correction and the major correction.
  2. Since we interpret the critic values in DiCGAN as the ranking scores, the relationship between the user preferences (reflected by ranking scores) and the distribution distance (represented by critic values) [Proposition 2 and Corollary 2.1] can be derived. This is the first time that such a relationship is rigorously shown.

F. Discussions about Pairwise Regularization to Generator

In this section, we claim that adding the pairwise regularization to the generator requires heavy supervision and is invalid. Our DiCGAN thus does not consider such regularization.

As the target is to learn the desired data distribution, the regularization on the generator can be used to make the critic values of the generated samples larger than those of the undesired samples. Specifically, a selector is first applied to give a full ranking for the training data, and then the bottom K_0 samples are picked up as the undesired samples. The pairwise preferences are then defined over the generated samples and the undesired samples. Note that the undesired subset of the training data requires labeling all training data.

We consider two cases of adding the regularization to the generator. First, we only add the pairwise regularization to the generator (PRG-1). Second, we add the regularization to the generator together with the regularization on the critic (PRG-2).

The objective for PRG-1 is as follows:

$$\begin{aligned} L_D &= \mathbb{E}_{p_{\mathrm{r}}(x)}\left[D(x)\right] - \mathbb{E}_{p_{\theta}(x)}\left[D\left(x\right)\right], \\ L_G &= \mathbb{E}_{p_{\theta}(x)}\left[D\left(x\right)\right] - \lambda_g \frac{1}{|\mathrm{S}'|}\sum_{s\in \mathrm{S}'}\left[h\left(s\right)\right], \end{aligned} \tag{15}$$

where $h(s)$ is Eq. (5). $\mathrm{S}^{'}$ is the pairwise preferences constructed between the generated data and the undesired data, i.e., $\mathrm{S}^{'} = \left\{s = (x_1, x_2) | x_1 \succ x_2, x_1 \sim p_{\theta}(x), x_2 \sim p_{\mathrm{u}}(x)\right\}$. Now the generator consists of two terms, the original WGAN loss on the generator aims to achieve $\mathbb{E}_{p_{\theta}(x)}\left[D\left(x\right)\right] > \mathbb{E}_{p_{\mathrm{r}}(x)}\left[D(x)\right]$, while the regularization aims to achieve $\mathbb{E}_{p_{\theta}(x)}\left[D\left(x\right)\right] > \mathbb{E}_{p_{\mathrm{u}}(x)}\left[D(x)\right]$. Since the undesired data is a subset of the real data, i.e., $\{x|x \sim p_{\mathrm{u}}(x)\} \subseteq \{x|x \sim p_{\mathrm{r}}(x)\}$, the WGAN loss always dominates the training of the generator. Therefore, PRG-1 degenerates to WGAN.

The objective for PRG-2 is as follows:

$$\begin{aligned} L_D &= \mathbb{E}_{p_{\mathrm{r}}(x)}\left[D(x)\right] - \mathbb{E}_{p_{\theta}(x)}\left[D\left(x\right)\right] - \lambda \frac{1}{|\mathrm{S}|}\sum_{s\in \mathrm{S}}\left[h\left(s\right)\right], \\ L_G &= \mathbb{E}_{p_{\theta}(x)}\left[D\left(x\right)\right] - \lambda_g \frac{1}{|\mathrm{S}'|}\sum_{s\in \mathrm{S}'}\left[h\left(s\right)\right], \end{aligned} \tag{16}$$

where S is constructed based on (4). Although the generator consists of two terms, the same as our analysis about PRG-1, the extra pairwise regularization on the generator is invalid. Meanwhile, the extra pairwise regularization on the critic works like that in DiCGAN. Therefore, the whole framework degenerates to DiCGAN.

IV. Case Study on Synthetic Data

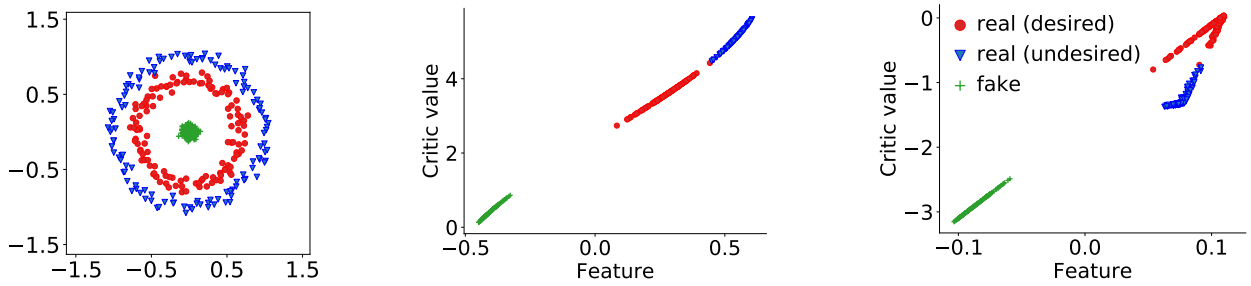
To gain an intuitive understanding of the differences between our DiCGAN and WGAN regarding the critic and the generator, we conduct a case study on a synthetic dataset.

The synthetic dataset consists of two concentric circles by adding Gaussian noise with a standard deviation of 0.05, which is a 2D mixture Gaussian distribution with two modes (See Fig. 4a). The samples located on the inner circle are considered to be the desired data, while the samples on the outer circle are defined as the undesired data. By labeling the desired data as $y = 1$ and the undesired data as $y = 0$, we can construct the pairwise preference for two samples x_1 and x_2 based on their labels. Namely, $x_1 \succ x_2$ if $y_1 = 1 \wedge y_2 = 0$, and vice versa. The pairs are constructed within each mini-batch. Our target is to learn the distribution of the desired data (i.e., samples on the inner circle), using the whole data along with the constructed pairwise preferences.

A. WGAN vs DiCGAN on Critic

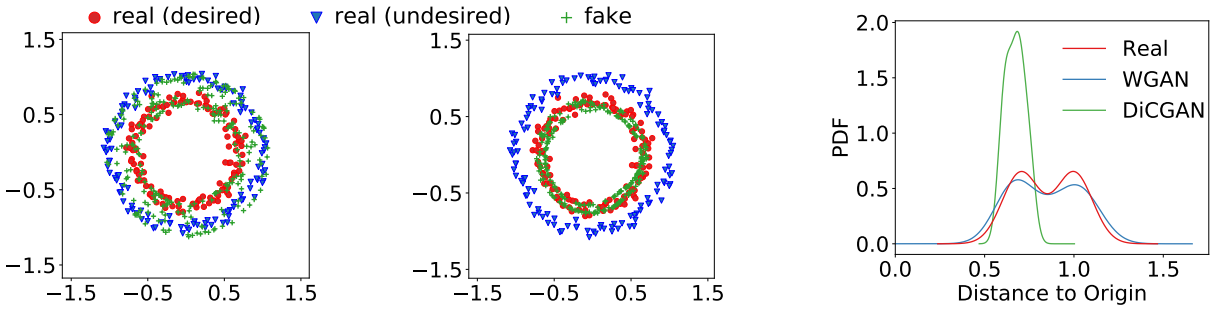
Experiment setting: we fix the generator and simulate the fake data as the 2D Gaussian blob with a standard deviation of 0.05 (green pluses). We first train the critic until convergence. Then, we project the output on the second last layer of the critic into 1D space using kernel principal components analysis (PCA), to obtain the projected features. To explore the difference between the critics of WGAN and DiCGAN, we draw the curve of the critic values versus the projected features for WGAN and DiCGAN, respectively (Fig. 4b, 4c).

From Fig. 4b, 4c, we can see: (1) in terms of the real data and the fake data, the critic of both WGAN and DiCGAN can achieve perfect discrimination. Meanwhile, the projected features of the real data and those of the fake data are also completely separated; (2) in terms of the real desired data and the real undesired data, the critic of DiCGAN assigns higher values to the desired samples, compared to the undesired samples. This is because our ranking loss expects a higher ranking score (i.e., critic value) for the desired sample. (3) In contrast, the critic of WGAN assigns lower values to the desired data since the desired data is closer to the fake data compared to the undesired data.



(a) Data (b) WGAN (c) DiCGAN

Fig. 4: Comparison of the critic in (b) WGAN and (c) DiCGAN. DiCGAN's critic can assign higher critic values for real desired data than real undesired data while WGAN's critic cannot. "Feature" is obtained by using kernel PCA to project the output on the second last layer of the critic into 1D space.



(a) WGAN (b) DiCGAN (c) PDF vs. distance

Fig. 5: (a-b) Visualization of the generated samples from WGAN and DiCGAN. The fake data is expected to overlap with the real desired data only. (c) Probability density function (PDF) vs. sample distance to the origin.

B. WGAN vs DiCGAN on Generator

Experiment setting: we train the critic and the generator following the regular GANs' training procedure. The generation results of WGAN and DiCGAN are shown in Fig. 5a, 5b.

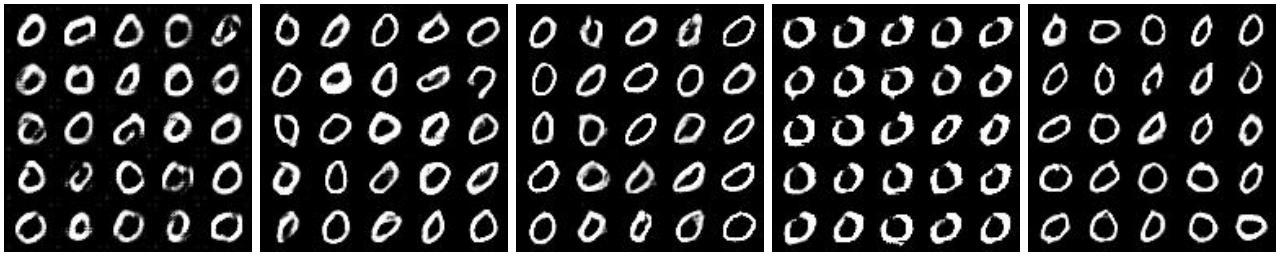
DiCGAN (shown in Fig. 5b) only generates the user-desired data. Namely, generated data covers the inner circle. In contrast, WGAN (shown in Fig. 5a) generates all data. Namely, generated data covers the inner circle and the outer circle. As the critic in DiCGAN can guide the fake data towards the real data region and away from the undesired data region, the generator thus produces data that is similar to the real desired data. Because the critic in WGAN pushes the fake data to the region of all real data, the generator finally produces the whole real-alike data.

Further, we calculate the distance from the real samples to the origin and plot the probability density function versus the distance in Fig. 5c. We also do this for the generated samples from WGAN and DiCGAN, respectively. It shows that DiCGAN only captures one mode of the real data distribution, consistent with the results that DiCGAN only produces desired samples. In contrast, WGAN captures all modes of the real data distribution, meaning that WGAN generates all real data.

V. Experimental Study

Our DiCGAN for desired data generation has various applications in the real world. In particular, we apply our DiCGAN to two applications: 1) generating images that meet the user's interest for a given dataset, which can be used for image search [8]. 2) optimizing biological products with desired properties, which can automate the process of designing DNA sequences for usage in medicine and manufacturing [7]. In these applications, we verify that our DiCGAN only using local knowledge (i.e., user preferences) outperforms current methods relying on global knowledge when labels of desired data are limited. Furthermore, we study the relation between critic values and user preferences as well as the effects of each component in DiCGAN.

Baselines We compare DiCGAN with WGAN [9], CWGAN [10], FBGAN [7] and GAN-FT. 1) WGAN is trained with only the desired data to derive the desired data distribution. 2) CWGAN is the extension of GAN with a conditional label c. To train CWGAN, we split the training data into the desired class ($c = 1$) and the undesired class ($c = 0$) based on global knowledge. Then $p(x|c = 1)$ is the desired data distribution. 3) FBGAN adopts an iterative training paradigm to derive the desired data distribution. First, FBGAN is pre-trained with all training data. At each training epoch, FBGAN resorts to an extra selector to select the desired samples from the generated samples and use them to replace the least-recently added samples in the training dataset. Then FBGAN performs regular GAN training with the updated training data. 4) GAN-FT is to fine-tune a pre-trained GAN with a classification loss on desired data. It is possible to use GAN loss defined between the generated data and the desired data to constrain the quality of desired data during the fine-tuning of GAN-FT. This is actually similar to the baseline WGAN that is trained on the desired subset of training data. Thus it would still suffer from poor data quality issues when there is limited desired data in the training dataset.



(a) WGAN (25/25) (b) CWGAN (25/25) (c) FBGAN (25/25) (d) GAN-FT (25/25) (e) DiCGAN (25/25)

Fig. 6: Generated images on MNIST by (a) WGAN, (b) CWGAN, (c) FBGAN, (d) GAN-FT and (e) DiCGAN

.

Iter 0 (1.0%) Iter 200 (16.1%) Iter 400 (28.2%) Iter 1000 (95.8%) Iter 2000 (99.9%)

Fig. 7: Generated images of DiCGAN on MNIST during the training process. DiCGAN learns the distribution of small digits, which gradually generates more small digit images. The % denotes the percentage of zero digits in $50K$ generated samples.

Datasets MNIST [23] consists of 28×28 images with digit zero to nine. $50K$ training images are regarded as training data. CelebA-HQ [24] is the high-quality subset of Celeb Faces Attributes Dataset, which has $30K$ face images of celebrities. We use all images as the training data and resize them to 64×64. The gene sequence dataset [7] contains $3,655$ gene sequences with a maximum length of 156 codings for proteins collected from the Uniprot database. All methods applied to the datasets use the same supervision for a fair comparison. On MNIST and CelebA-HQ, we resort to class labels to derive the desired data distribution. On the gene sequence dataset, we resort to an analyzer that can evaluate the desired property for genes to derive the desired data distribution.

Remark 2. *Considering pairwise preferences over explicitly labeling what the user considers to be good data or not is beneficial especially given the limited supervision, which will be verified in the following experiments.*

Evaluation Metric: To evaluate the performance of learning the desired data distribution, we calculate the percentage of desired data (PDD) in GAN's generation. PDD $= \frac{|\{x|x \text{ is desired}, x \in \mathrm{X_g}\}|}{|\mathrm{X_g}|} \times 100\%$, where $\mathrm{X_g}$ are generated samples.

A. Capturing Small Digits on MNIST

Suppose the user is interested in learning the distribution of small digits on MNIST. Zero is the smallest digit of MNIST, thus as the desired data.

Networks & Hyperparameters By a coarse grid search, the balance factor λ is set to 1. The ranking margin m is set to 1 following [25]. The batch size b is set to 50. The network architecture of the critic and generator in our DiCGAN are based on WGAN-GP [18]. See Supplementary for details. The baselines share the same architecture for a fair comparison. The optimizer is Adam [26] with a learning rate of $1e$-4 and $\beta_1 = 0.5, \beta_2 = 0.9$. The number of critic iterations per generator iteration n_{critic} is 5.

Training As for WGAN and CWGAN, zero digits in the training data are regarded as the desired samples ($c = 1$), whose size is $4,950$. The other digits are labeled as the undesired samples, whose size is $45,050$ ($c = 0$). WGAN is only trained with the desired data. CWGAN conditions on c to model a conditional data distribution $p(x|c)$ for MNIST. For GAN-FT, we first pre-trained WGAN-GP with all digit images. Then we fine-tuned its generator with a classifier loss that makes the generated samples classified as digit zero. FBGAN and our DiCGAN both introduce the generated samples into the training dataset during the training. The labels of the generated samples are obtained by resorting to a classifier, pre-trained for digit classification. At every training epoch, FBGAN generates $50K$ samples and requests the classifier to label them. Then the selector in FBGAN will rank the images using their corresponding labels, where the smaller digits are ranked higher. The selector selects the generated images with digits ranked in the top 50%, i.e., small digits, as the desired data to replace old training data. As for DiCGAN, the pairwise comparison can be obtained for two images x_1 and x_2 according to their predicated label y_1 and y_2, namely $x_1 \succ x_2$ if $y_1 < y_2$, and vice versa. At each iteration, #pairwise preferences n_s is 25. #iteration per minor correction $n_{\mathrm{i}} = 200$.

TABLE III: Percentage of desired data in the generation (PDD) of various GANs on MNIST. Best results are highlighted in bold. Top 1 means digit zero. Top 5 means digits zero to four.

Method	Original	WGAN	CWGAN	FBGAN	GAN-FT	DiCGAN
Top 1	9.9	97.3	95.0	**100.0**	**100.0**	**100.0**
Top 5	51.1	98.2	96.4	**100.0**	**100.0**	**100.0**

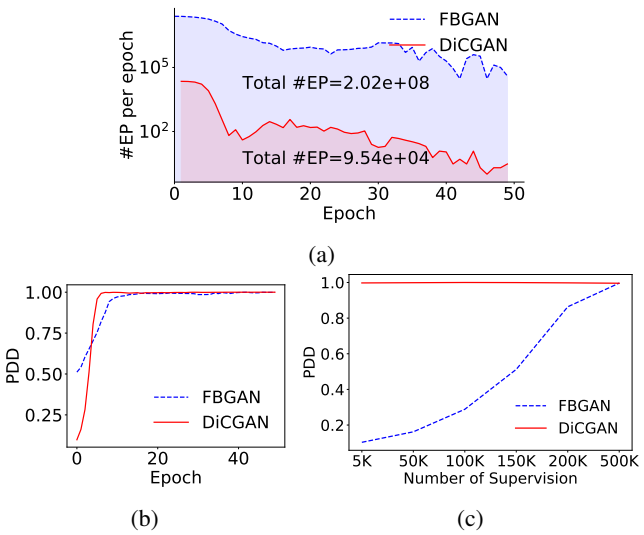


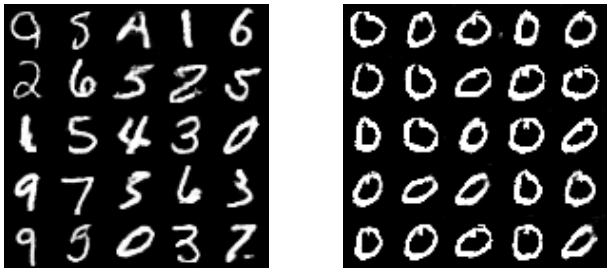
Fig. 8: Comparison of DiCGAN and FBGAN on MNIST. (a) plots used #EP per epoch. (b) plots PDD versus the training epoch. (c) plots PDD versus the number of supervision.

#generated samples for each minor correction $n_{\mathrm{g}} = 50K$.

Fig. 7 presents the generated MNIST images randomly sampled from the generator of DiCGAN. It shows that the generated MNIST digits gradually shift to smaller digits during the training, and converge to the digit zero. For each method, we sample $50K$ samples from the generator and calculate the percentage of digit zero and digits zero to four among the generated digits for quantitative evaluation. In Table III, only small digits are generated by DiCGAN and FBGAN; WGAN and CWGAN can also learn the distribution of the desired digit since the dataset is simple and has relatively sufficient data for the desired digit. The visual results shown in Fig. 6 are consistent with the quantitative results. However, when the dataset is complex and the desired data is insufficient, WGAN and CWGAN fail, which is described in Sect. V-B. GAN-FT also only generates digit zero, but it suffers from mode collapse problem. The generated images have low diversity (Fig. 6d). This is because there lacks data quality guarantee during the later fine-tuning stage.

1) Comparison of DiCGAN and FBGAN: Though FBGAN achieves good performance in learning the desired data distribution, it requires a lot of supervision information from the selector. We calculate the number of effective pairs (#EP) used in DiCGAN and FBGAN, respectively. #EP in DiCGAN denotes the total number of explicitly constructed pairs during the training, i.e., #EP $= \sum_{i=1}^{n_{\mathrm{e}}} \sum_{j=1}^{n_{\mathrm{i}}} n_{\mathrm{s}}$. As for FBGAN, its selector ranks all generated samples and selects the desired samples from them at each epoch. Therefore, #EP can be induced by the implicit pairs implied by the desired generated samples versus the undesired generated samples, i.e., #EP $= \sum_{i=1}^{n_{\mathrm{e}}} n_{\mathrm{gd}} \times n_{\mathrm{gu}}$, where n_{e} is the number of training epochs. where n_{gd} and n_{gu} denote the number of desired samples and undesired samples in the generation, respectively.

Fig. 8a plots FBGAN's and DiCGAN's used #EP at each epoch, respectively. It shows that (1) the #EP used in DiCGAN is much smaller than that in FBGAN at each training epoch;



(a) FBGAN (b) DiCGAN

Fig. 9: The generated results of (a) FBGAN and (b) DiCGAN on MNIST given limited supervision.

(2) the total #EP used in DiCGAN is significantly less than that in FBGAN, which can be reflected from the shadow area. In total, DiCGAN used $9.53e4$ effective pairs while FBGAN used $2.02e8$ effective pairs. Our DiCGAN is scalable to the large training dataset, e.g. MNIST. #EP in DiCGAN is linearly correlated to the training size. In contrast, #EP in FBGAN is determined by n_{gd} and n_{gu}, which are both linearly correlated to the training size. Thus, #EP in FBGAN is quadratically correlated to the training size.

We plot the ratio of digit zero in the generated data (PDD) of DiCGAN and FBGAN during the training process in Fig. 8b. It shows that DiCGAN converges faster than FBGAN.

2) Comparing DiCGAN and FBGAN given the limited supervision: We conduct the experiment on MNIST. Specifically, the query amount of resorting to the pre-trained classifier to obtain the prediction of the generated samples is restricted to $5K$ for both FBGAN and DiCGAN.

Table IV shows that DiCGAN can learn the desired data distribution, generating 99.7% zero digits, while FBGAN fails, generating 10.3% digit zero, which is consistent with the visual results in Fig. 9a and Fig. 9b.

TABLE IV: PDD on MNIST given limited supervision.

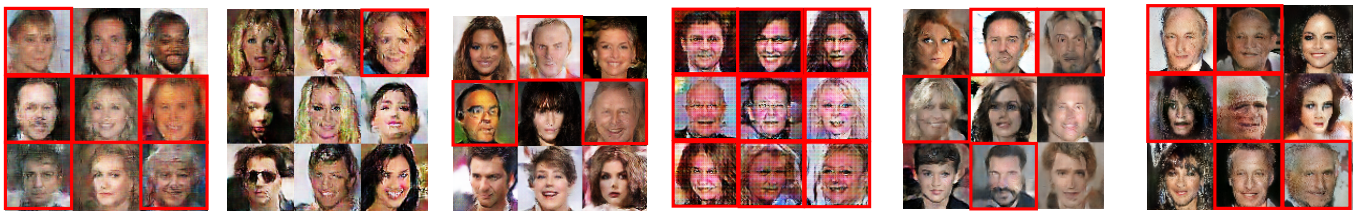
Method	Top 1	Top 5
FBGAN	10.3	52.6
DiCGAN	**99.7**	**99.9**

We explore the gap in the performance between DiCGAN and FBGAN evolves as the number of supervision increases on MNIST. Specifically,the query amount of resorting to the pre-trained classifier to obtain the prediction of the generated samples is restricted to $5K, 50K, 100K, 150K, 200K, 500K$ for both FBGAN and DiCGAN.

Fig. 8c plots PDD versus the number of supervision for FBGAN and DiCGAN, respectively. It shows that (1) DiCGAN always learns the desired data distribution even given the limited supervision; (2) when given the limited supervision, FBGAN fails to learn the desired data distribution, i.e., achieving a small PDD; (3) FBGAN performs better and achieves a higher PDD, narrowing the performance gap with DiCGAN as the number of supervision increases.

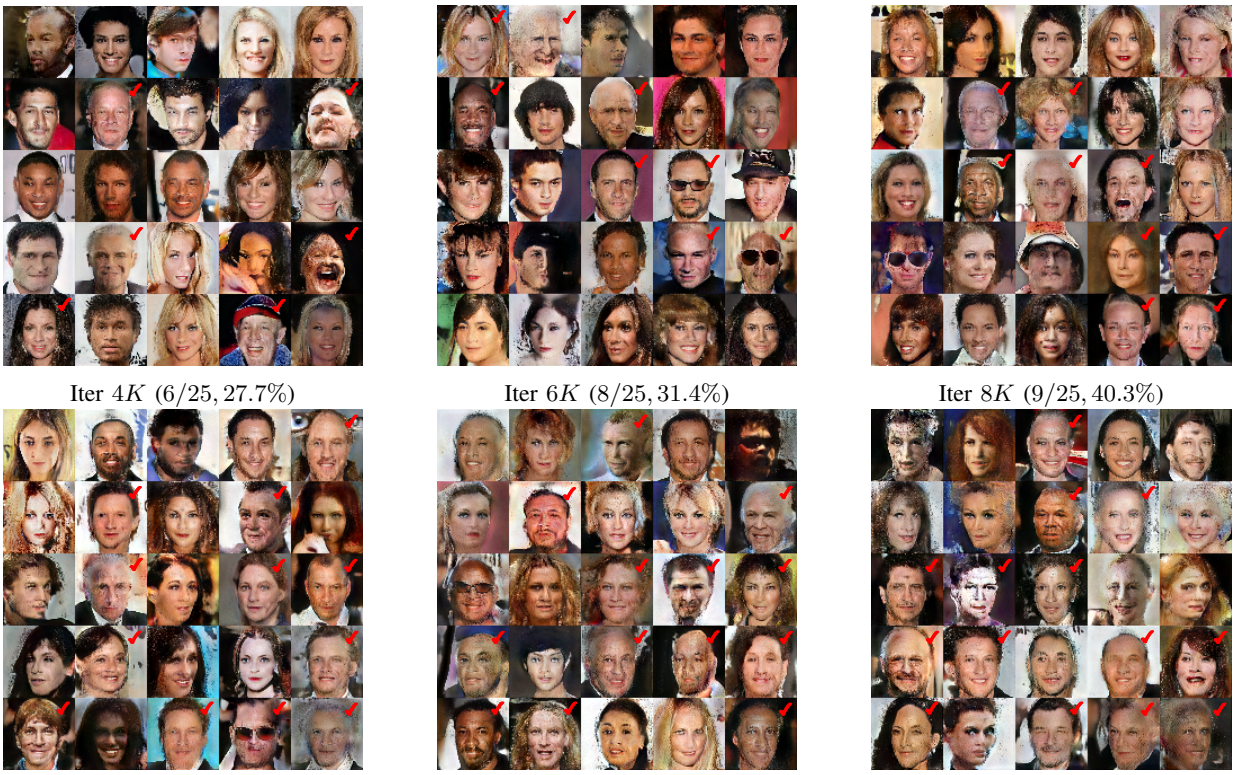
B. Capturing Old Face Images on CelebA-HQ

Suppose the user is interested in learning the distribution of old face images on CelebA-HQ.



(a) WGAN (6/9) (b) CWGAN (1/9) (c) FBGAN (3/9) (d) GAN-FT (9/9) (e) DiCGAN (4/9) (f) DiCGAN$_{\text{style}}$ (6/9)

Fig. 10: Generated images on CelebA-HQ by (a) WGAN, (b) CWGAN, (c) FBGAN, (d) GAN-FT, (e) DiCGAN and (e) DiCGAN$_{\text{style}}$. The red boxes refer to the images which are classified as old images.



Iter $4K$ $(6/25, 27.7\%)$ Iter $6K$ $(8/25, 31.4\%)$ Iter $8K$ $(9/25, 40.3\%)$

Iter $10K$ $(12/25, 46.0\%)$ Iter $12K$ $(13/25, 51.2\%)$ Iter $14K$ $(14/25, 57.0\%)$

Fig. 11: Generated images of DiCGAN$_{\text{style}}$ on CelebA-HQ during the training process. DiCGAN$_{\text{style}}$ learns the distribution of old faces, which gradually generates more old face images. The red ticks refer to the images which are classified as old images. The % denotes the percentage of old faces in $50K$ generated samples.

Networks & Hyperparameters The balance factor λ and the ranking margin m is set to 1. The batch size b is set to 64. The network architecture of the critic and generator in our DiCGAN are based on WGAN-GP [18]. See Appendix for details. The baselines share the same architecture for a fair comparison. The optimizer is Adam with a learning rate of $2e$-4 and $\beta_1 = 0.5, \beta_2 = 0.999$. n_{critic} is set to 5. Further, we use an advanced GAN architecture, StyleGAN (*https://github.com/NVlabs/stylegan2*) [27] to implement our DiCGAN, denoted as DiCGAN$_{\text{style}}$. The networks are optimized with Adam with $\beta_1 = 0, \beta_2 = 0.9$. The generator G's learning rate is $1e$-4 while the critic D's is $3e$-4 [28]. n_{critic} is set to 1.

Training There are $6,632$ old face images, labeled as desired, and $23,368$ young face images, labeled as undesired, in the training data. WGAN is only trained with the constructed desired dataset. CWGAN conditions on c to model a conditional data distribution $p(x|c)$. A classifier, pre-trained for classifying young faces and old faces, is adopted for predicting the labels for the generated face images. Particularly, the query amount of resorting to the classifier is restricted to $30K$. As for FBGAN, at every training epoch, FBGAN generates $5K$ images, and those classified as old faces are selected by the selector to replace the old training data. As for GAN-FT, we first pre-trained WGAN-GP with all images including young faces and old faces. Then we fine-tuned the pre-trained generator with the classifier loss that makes the generated samples classified as old faces. As for DiCGAN, the generated face image classified as an old face is preferred over the face image classified with the young attribute. At each iteration, n_s is set to 64. n_{i} is set to $1K$. n_{g} is set to $1K$. As for DiCGAN$_{\text{style}}$, at each iteration, n_s is set to 64. n_{i} is set to 500. n_{g} is set to $30K$.

We visualize the generated face images randomly sampling from the generator of each model in Fig. 10. For each model,

TABLE V: Percentage of desired data in the generation (PDD) and image quality (FID) of various GANs on CelebA-HQ. The best results are highlighted in bold. The second best results are underlined. The strikethrough on PDD of WGAN and GAN-FT denotes that they suffer from severe low-quality issues (large FID), generating very blur face images (Fig. 10) and thus its PDD is not really meaningful.

Method	Original	WGAN	CWGAN	FBGAN	GAN-FT	DiCGAN	DiCGAN$_{\text{style}}$
PDD	22.1	~~**76.0**~~	8.3	24.7	~~**99.7**~~	<u>33.4</u>	**57.0**
FID	-	115.4	79.7	51.6	107.1	<u>49.7</u>	**36.5**

we sample $50K$ samples from the generator and then calculate the percentage of old face images (PDD) and the image quality score, i.e., Frechet Inception Distance (FID) among the generated samples for quantitative evaluation in Table V. From Fig. 10 and Table V, (1) though WGAN mainly generates desired data, it has poor generation since its training data only consists of the desired subset, and thus is insufficient, which has only $6,632$ face images. The generated face images are blurred. Meanwhile, WGAN's FID score is the highest among all methods, i.e., 115.4, quantitatively showing the poorest generation quality. (2) CWGAN has better generation quality than WGAN as it is trained with sufficient training data, $30K$ samples, but fails to shift towards the desired data distribution. There is only one old face image out of 9 randomly sampled images in the visualization result. Its PDD (8.3%) is smaller than the training data (Original, 22.1%). This is because the undesired data, i.e, the majority in the training data, dominates the generation of CWGAN. (3) FBGAN can achieve relatively good quality, with relatively small FID, but only slightly shift towards the distribution of desired data (PDD=24.7%) due to limited supervision. (4) GAN-FT almost generates old faces, but the quality is poor, verified by a large FID quality score and low-quality visual results in Fig. 10d. (5) DiCGAN achieves the best image quality among all the methods. Its FID score is the lowest. In addition, DiCGAN shifts more towards the desired data distribution than CWGAN and FBGAN, proven by a larger PDD.

On the other hand, WGAN-GP architecture is limited to approximating the complex distribution of CelebA-HQ data and thus cannot generate images with very high quality. Then, the introduction of generated samples into the training data will degrade the quality of generation. Therefore, DiCGAN implemented with WGAN-GP architecture is restricted with certain amounts of minor corrections and data replacement in order to obtain a good quality, achieving relatively low PDD. This problem can be improved by introducing a more advanced GAN architecture, StyleGAN. DiCGAN$_{\text{style}}$ can conduct more minor corrections and use more generated data to replace the training data, finally making the training data distribution shift very close to the desired data distribution. Thus, DiCGAN$_{\text{style}}$'s generation contains more desired samples than DiCGAN, i.e., larger PDD in Table V. Meanwhile, the generation has good quality with the best FID. We present generated images of DiCGAN$_{\text{style}}$ during the training process in Fig. 11. There gradually appears more desired face images, i.e., old face images in DiCGAN$_{\text{style}}$'s generation.

Fig. 12: Nearest neighbors of generated desired images in the training dataset. The distance is measured by the ℓ_2 distance between images. Images on the left of the red vertical line are samples generated by our DiCGAN. Images on the right are top 5 nearest neighbors in the training dataset.

Fig. 12 shows the nearest neighbors of generated old images in the training dataset (given old face images), which demonstrates that our DiCGAN is not simply memorizing training images, but generates novel desired images. Thus, DiCGAN can perform data augmentation for desired samples.

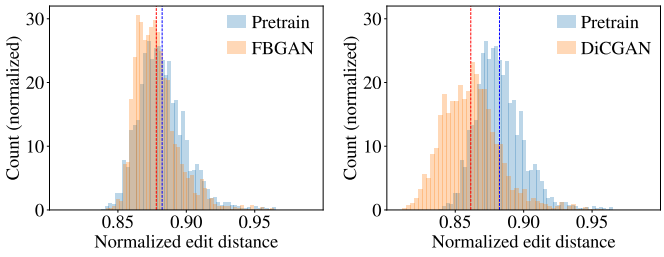
C. Simulating Synthetic Genes with Antimicrobial Properties

Consider the biologist is interested in designing genes coding for antimicrobial peptides (AMPs), which are peptides with broad antimicrobial activity against bacteria, viruses, and fungi [29]. We can apply our DiCGAN to help optimize the gene coding for AMPs from an existing gene sequence dataset [7]. Namely, our target is to learn the distribution of genes coding for AMPs on the gene sequence dataset.

Networks & Hyperparameters Both the balance factor λ and the ranking margin m are set to 1. The batch size b is set to 64. All methods are implemented with the networks as FBGAN [7]. The code of the network architecture can be found on FBGAN's official implementation (*https://github.com/av1659/fbgan*). The networks are optimized with Adam with a learning rate of $1e$-4 and $\beta_1 = 0.5, \beta_2 = 0.9$. n_{critic} is set to 10.

Training There is no labeling about whether the genes have the desired property in the training data. Therefore, WGAN, CWGAN, and GAN-FT cannot be applied. Following FBGAN [7], we resort to an analyzer that can evaluate the property for genes. We pertain FBGAN and DiCGAN as vanilla WGAN using $3,655$ gene sequences. Then, we train FBGAN and DiCGAN with $2K$ gene sequences and collect the results for each method. Here we limit the amount of querying the analyzer to $6K$. n_{i} is set 31. n_{g} is set to 500.

Sample selection in FBGAN The selector in FBGAN selects desired samples based on the evaluation of the analyzer, which is able to predict the probability of a gene coding for AMPs. Specifically, the analyzer first estimates the probability of generated genes coding for AMPs. Then, the generated genes with the estimated probability over 0.8, considered as the desired genes, are selected by the selector to replace the old training data.



(a) FBGAN (b) DiCGAN

Fig. 13: Comparison of (a) FBGAN and (b) DiCGAN on the gene sequence dataset. The dashed line denotes the mean value. The normalized edit distance is calculated between synthetic proteins and real desired proteins. A smaller distance denotes the generated genes are more similar to the desired genes.

Pairwise preferences construction in DiCGAN We consider the analyzer [1] as the user, where a larger predicted value denotes that the gene is preferred for coding AMPs. Then, the pairwise comparison can be obtained for a pair of samples x_1 and x_2 according to their predicated values $p(x_1)$ and $p(x_2)$, i.e., $x_1 \succ x_2$ if $p(x_1) > p(x_2)$, and vice versa. At each iteration, n_s is set to 64. n_{i} is set to 31.

The difference between genes is evaluated via the Normalized Edit Distance (normalized Levenstein distance, NED) between the proteins coded by the corresponding genes [7]. The similarity of a generated gene to the desired gene can be evaluated using the averaged NED between its corresponding protein and all real desired proteins (i.e., AMPs). Particularly, a smaller NED w.r.t. AMPs denotes the generated genes more similar to genes coded for AMPs.

In Fig. 13, we compare DiCGAN and WGAN w.r.t. the NED between AMPs and the synthetic proteins. It shows that the synthetic proteins of DiCGAN shift toward a lower edit distance from AMPs, compared to the pretraining stage, i.e., WGAN. It means more genes coded for AMPs are generated by DiCGAN. However, FBGAN fails to shift its distribution towards the distribution of genes coding for AMPs.

TABLE VI: Percentage of desired data in the generation (PDD) and gene quality (%VG) of various GANs on the gene sequence dataset. The best results are highlighted in bold.

Method	FBGAN	DiCGAN
PDD	29.0	**98.8**
%VG	57.8	**67.0**

Further, we sample $50K$ genes from the generator for quantitative evaluation, which is collected in Table VI. The generated genes with the probability of coding for AMPs over 0.8 is considered as the desired genes. Then, the percentage of the desired genes among all $50K$ generated genes (PDD) is calculated. Particularly, almost all genes generated by DiCGAN can be classified as the desired genes, i.e., 98.8%. In contrast, FBGAN generates 29.0% desired genes. DiCGAN can learn the distribution of desired genes. However, FBGAN fails to derive the desired data distribution due to limited supervision.

[1]Or we can ask biological experts to compare pairs of samples in terms of the desired property if the desired properties cannot be expressed objectively.

On the other hand, we calculate the percentage of valid genes (%VG) [2]. The %VG of DiCGAN is 67.0% while that of FBGAN is 57.8%, which clarifies our methods achieve better quality than FBGAN. Their quality degradation compared to pre-trained WGAN is due to the introduction of generated genes as training data.

D. Study on critic values versus user preferences

We apply DiCGAN to evaluate the critic values for all undesired data and all desired data on MNIST, CelebA-HQ, and the gene sequence dataset, respectively. Then we calculate the mean critic values for desired data and undesired data, respectively, with 95% confidence interval. Meanwhile, we conduct the two-sample one-sided t-Test [30] for their mean critic values under the null hypothesis of equal means and the alternative hypothesis that the mean of the desired data is greater than that of the undesired data.

TABLE VII: The mean (with 95% confidence interval) and the two-sample one-sided t-Test results of critic values for desired data and undesired data on MNIST, CelebA-HQ, and the gene sequence dataset.

Dataset	mean critic value		p value
	desired	undesired	(desired VS. undesired)
MNIST	1.5 ± 0.01	0.2 ± 0.01	0.00
CelebA-HQ	0.9 ± 0.02	0.4 ± 0.01	$2.58e$-285
gene	8.9 ± 0.14	8.1 ± 0.05	$3.54e$-24

The results in Table VII show that the average critic value for desired data is significantly larger than that of undesired data with a very small p-value. Therefore, it verifies the claim (mentioned in Sect. III-B) that the ranking loss can encourage high critic values to be assigned to the real desired data while low critic values are assigned to real undesired data.

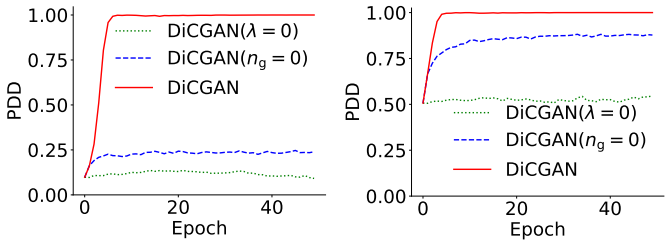
E. Ablation Study

The objective in our DiCGAN (Eq. (6)) consists of two components, i.e., the WGAN loss, which serves as the cornerstone of DiCGAN, and the ranking loss, which serves as the correction to WGAN. Meanwhile, we introduce the operation of replacement (Eq. (10)) during the model training.

To analyze the effects of the correction for WGAN (the third term in Eq. (6)) and the replacement operation, we plot the percentage of desired data in the generation (PDD) versus the training epoch for DiCGAN ($\lambda = 0$), DiCGAN ($n_{\text{g}} = 0$) and DiCGAN in Fig. 14a, 14b. Meanwhile, the converged percentage of desired samples (PDD) is reported in Fig. 14c.

1) **Without the correction term ($\lambda = 0$), DiCGAN cannot learn the desired data distribution.** The PDD of DiCGAN ($\lambda = 0$) remains constant during training on MNIST (Fig. 14a, 14b) compared with that of the original dataset (Fig. 14c). This is because the WGAN term in DiCGAN ($\lambda = 0$) focuses on learning the training data distribution.

[2]Correct gene structure is defined as a string starting with the canonical start codon "ATG", followed by an integer number of codons of length 3, and ending with one of three canonical stop codons ("TAA", "TGA", "TAG") [7].

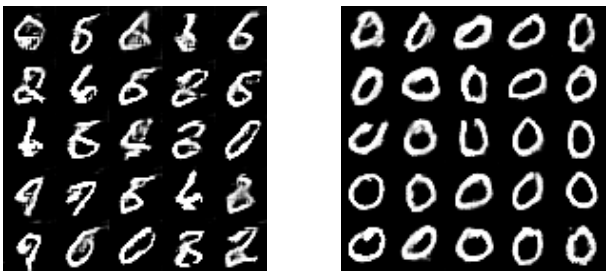


(a) top 1 (b) top 5

Method	top 1	top 5
Original	9.9	51.1
DiCGAN ($\lambda = 0$)	9.1	54.6
DiCGAN ($n_g = 0$)	24.0	87.8
DiCGAN	**100.0**	**100.0**

(c) PDD

Fig. 14: Ablation study on MNIST. (a-b) PDD vs. epoch in the generation of DiCGAN ($\lambda = 0$), DiCGAN ($n_g = 0$) and DiCGAN. (c) PDD in the data from the original dataset, DiCGAN ($\lambda = 0$), DiCGAN ($n_g = 0$) and DiCGAN.



(a) PRG-1 (PDD=13.9) (b) PRG-2 (PDD=99.4)

Fig. 15: Generated digits and PDD of (a) PRG-1 & (b) PRG-2.

2) **Without the replacement ($n_g = 0$), DiCGAN makes a minor correction to the generated distribution.** In Fig. 14a, 14b, the PDD of DiCGAN ($n_g = 0$) slightly increases compared with the original dataset. This is consistent with our analysis that the correction term would drive the generation towards the desired data distribution.
3) **DiCGAN learns the desired data distribution with a sequential minor correction.** The PDD of DiCGAN grows with training and reaches almost 100% when convergence. The correction term drives DiCGAN's generation towards the desired data slightly at each epoch. With the iterative replacement, the minor correction sequentially accumulates, and finally the generated distribution shifts to the desired data distribution.

F. Pairwise regularization on the Generator

As discussed in Sect. III-F, the pairwise regularization is possibly added to the generator. We consider two cases of adding the regularization to the generator. First, we only add the pairwise regularization to the generator (PRG-1). Second, we add the regularization to the generator together with the regularization on the critic (PRG-2).

We conducted experiments on MNIST to show the effectiveness of these two methods. λ and λ_g are both set to 1. As shown in Fig. 15, PRG-1 failed to learn the desired data distribution. PRG-2 can learn the desired data distribution. The quantitative results are consistent with the visual results, with 13.9% and 99.4% PDD, respectively.

VI. Conclusions and Discussions

This paper proposes DiCGAN to learn the distribution of the user-desired data from the entire dataset using the pairwise preferences. This is the first work to promote the ratio of the desired data by incorporating user preferences directly into the data generation. We empirically demonstrate the efficacy of DiCGAN in two real-world applications – generating images that meet the user's interest for a given dataset and optimizing biological products with desired properties. Especially, our DiCGAN outperforms baselines in the cases of insufficient desired data and limited supervision.

Though it is superior to existing methods in terms of desired data generation when there is insufficient desired data, our DiCGAN cannot handle the case when there are extremely limited desired data, e.g., few-shot even one shot. Furthermore, as shown in our experimental study, high-resolution high-quality desired image generations require an advanced GAN architecture, which incurs heavy computational costs. There is an ongoing research direction of GAN that aims to generate high-resolution high-quality data with light architecture designs, which can mitigate such a limitation.

Acknowledgment

YP and IWT are supported by A*STAR CFAR. IWT is also supported by Australian Research Council under grants DP200101328. YY and XY are supported by the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531) and the Program for Guangdong Provincial Key Laboratory (Grant No. 2020B121201001).

Appendix

Hyperparameter setting and network structures.

For the setup of the balance factor λ, we set λ as $0.1, 0.5, 1, 5, 10$, respectively and found that $\lambda = 1$ consistently performs well on all datasets. Thus we set $\lambda = 1$ for all datasets.

TABLE VIII: The architecture of our critic for MNIST.

$x \in \mathbb{R}^{1\times 28\times 28}$
Conv2d 5×5, stride 2, pad 2, $1 \to 64$; ReLU
Conv2d 5×5, stride 2, pad 2, $64 \to 128$; ReLU
Conv2d 5×5, stride 2, pad 2, $128 \to 256$; ReLU
linear, $256 \times 4 \times 4 \to 1$

TABLE IX: The architecture of our generator for MNIST.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
linear, $128 \to 256 \times 4 \times 4$
ConvTranspose2d 5×5, stride 1, pad 0, $256 \to 128$; ReLU
ConvTranspose2d 5×5, stride 1, pad 0, $128 \to 64$; ReLU
ConvTranspose2d 8×8, stride 2, pad 0, $64 \to 1$; Sigmoid

TABLE X: The architecture of our critic for CelebA-HQ.

$x \in \mathbb{R}^{3\times64\times64}$
Conv2d 5×5, stride 2, pad 2, $1 \rightarrow 64$; LeakyReLU
Conv2d 5×5, stride 2, pad 2, $64 \rightarrow 128$; InstanceNorm2d; LeakyReLU
Conv2d 5×5, stride 2, pad 2, $128 \rightarrow 256$; InstanceNorm2d; LeakyReLU
Linear, $256 \times 8 \times 8 \rightarrow 1$

TABLE XI: The architecture of our generator for CelebA-HQ.

$z \in \mathbb{R}^{100} \sim \mathcal{N}(0, I)$
Linear, $100 \rightarrow 512 \times 4 \times 4$
ConvTranspose2d 5×5, stride 2, pad 2, output pad 1, $512 \rightarrow 256$; BatchNorm2d; ReLU
ConvTranspose2d 5×5, stride 2, pad 2, output pad 1, $256 \rightarrow 128$; BatchNorm2d; ReLU
ConvTranspose2d 5×5, stride 2, pad 2, output pad 1, $128 \rightarrow 64$; BatchNorm2d; ReLU
ConvTranspose2d 8×8, stride 2, pad 2, output pad 1, $64 \rightarrow 3$; Tanh

REFERENCES

[1] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[2] Z. Zhou, Y. Guo, and Y. Wang, "Handheld ultrasound video high-quality reconstruction using a low-rank representation multipathway generative adversarial network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 575–588, 2021.

[3] L. Yi and M.-W. Mak, "Improving speech emotion recognition with adversarial data augmentation network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 1, pp. 1–13, 2020.

[4] M. Yang, C. Li, Y. Shen, Q. Wu, Z. Zhao, and X. Chen, "Hierarchical human-like deep neural networks for abstractive text summarization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 6, pp. 2744–2757, 2021.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative Adversarial Nets," in *NeurIPS*, 2014.

[6] N. Killoran, L. J. Lee, A. Delong, D. Duvenaud, and B. J. Frey, "Generating and designing dna with deep generative models," *arXiv preprint arXiv:1712.06148*, 2017.

[7] A. Gupta and J. Zou, "Feedback gan for dna optimizes protein functions," *Nature Machine Intelligence*, vol. 1, no. 2, p. 105, 2019.

[8] Z. Yu and A. Kovashka, "Syntharch: Interactive image search with attribute-conditioned synthesis," in *CVPR Workshops*, 2020.

[9] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[10] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[11] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *NeurIPS*, 2017.

[12] T. Lu and C. Boutilier, "Learning mallows models with pairwise preferences," in *ICML*, 2011.

[13] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard GAN," in *ICLR*, 2019.

[14] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *ICML*, 2017.

[15] Y. Pan, B. Han, and I. W. Tsang, "Stagewise learning for noisy k-ary preferences," *Machine Learning*, vol. 107, no. 8-10, pp. 1333–1361, 2018.

[16] Y. Pan, I. W. Tsang, W. Chen, G. Niu, and M. Sugiyama, "Fast and robust rank aggregation against model misspecification," *Journal of Machine Learning Research*, vol. 23, no. 23, pp. 1–35, 2022.

[17] T.-Y. Liu, "Learning to rank for information retrieval," *Found. Trends Inf. Retr.*, vol. 3, no. 3, p. 225–331, 2009.

[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *NeurIPS*, 2017.

[19] A. Jolicoeur-Martineau, "On relativistic f-divergences," *arXiv preprint arXiv:1901.02474*, 2019.

[20] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *ICML*, 2005.

[21] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *ICML*, 2007.

[22] K. Zhou, G.-R. Xue, H. Zha, and Y. Yu, "Learning to rank with ties," in *ACM SIGIR*, 2008.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[24] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *ICLR*, 2018.

[25] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon, "Adapting ranking svm to document retrieval," in *ACM SIGIR*, 2006.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[27] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *CVPR*, 2020.

[28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *NeurIPS*, 2017.

[29] A. Izadpanah and R. L. Gallo, "Antimicrobial peptides," *Journal of the American Academy of Dermatology*, vol. 52, no. 3, pp. 381–390, 2005.

[30] B. L. Welch, "The generalization of 'student's'problem when several different population varlances are involved," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.