IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 12, 2021, accepted January 19, 2021, date of publication January 26, 2021, date of current version February 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3054822

Constrained Generative Adversarial Networks

XIAOPENG CHAO[1], JIANGZHONG CAO[1], YUQIN LU[1], QINGYUN DAI[2], AND SHANGSONG LIANG[3]

[1]School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China
[2]School of Electronic and Information Engineering, Guangdong Polytechnic Normal University, Guangzhou 510665, China
[3]School of Computer Science and Technology, Sun Yat-sen University, Guangzhou 510006, China

Corresponding author: Jiangzhong Cao (cjz510@gdut.edu.cn)

This research was supported by the National Natural Science Foundation of China (No. U1701266, No. 61906219), the Natural Science Foundation of Guangdong Province (No. 2018A030313751), the Science and Technology Planning Project of Guangdong Province (No. 2018B030322016), the Guangzhou Science and Technology Plan Project (no. 202002030386), and the Guangdong Higher Education Engineering Technology Research Center for Big Data on Intellectual Property of Manufacturing.

ABSTRACT Generative Adversarial Networks (GANs) are a powerful subclass of generative models. Yet, how to effectively train them to reach Nash equilibrium is a challenge. A number of experiments have indicated that one possible solution is to bound the function space of the discriminator. In practice, when optimizing the standard loss function without limiting the discriminator's output, the discriminator may suffer from lack of convergence. To be able to reach the Nash equilibrium in a faster way during training and obtain better generative data, we propose constrained generative adversarial networks, GAN-C, where a constraint on the discriminator's output is introduced. We theoretically prove that our proposed loss function shares the same Nash equilibrium as the standard one, and our experiments on mixture of Gaussians, MNIST, CIFAR-10, STL-10, FFHQ, and CAT datasets show that our loss function can better stabilize training and yield even better high-quality images.

INDEX TERMS Generative adversarial networks, Nash equilibrium, Lipschitz constraint.

I. INTRODUCTION

GANs (Generative Adversarial Networks [1]) which work with a minimax game consisting of a discriminative network D and a generative network G, are a subclass of generative models with implicit density, and have a variety of successful applications such as speech synthesis [2], super-resolution [3], [4], image inpainting [5]–[7] and image-to-image translation, etc [8]–[13]. The generator network maps a source of noise to the data space in order to generate samples from the real data distribution, while the discriminator one estimates the probability of the input data being real and thus discriminates between real and fake samples.

The goal of training GANs is to find the Nash equilibrium in the game such that the generator is able to recover the real data distribution exactly. When Nash equilibrium is reached, the output of the discriminator is supposed to converge. Typically, the training of GANs is performed by gradient descent techniques that are not designed to find Nash equilibrium, and may fail to converge [14]. Some techniques aim at reaching the convergence of discriminator, but most of them require very strong assumptions that are hard to satisfy in practice [14]–[16].

Lipschitz-based methods indirectly restrict the output of the discriminator, aiming to stabilize the training of GANs, which do not significantly help the convergence of the discriminator. For instance, WGAN [17] uses weight clipping to enforce a Lipschitz constraint on the discriminator. WGAN-GP [18] uses gradient penalty to confine the discriminator within the space of 1-Lipschitz functions. SNGAN [19] uses spectral normalization to make sure that the discriminator satisfies Lipschitz constraint.

Relativistic GANs [20] has introduced the ''relativistic discriminator'' to guarantee that the output of the discriminator for real data is decreasing when optimizing the generator, which is a key property missing from the original loss function proposed in [1]. If the output of the discriminator is not effectively limited, optimizing the original loss function can easily lead to gradient explosion.

Inspired by Nash equilibrium, we argue that a critical component related to Nash equilibrium is missing to help GANs better reach Nash equilibrium.

Accordingly, in this paper, we propose **C**onstrained **G**enerative **A**dversarial **N**etworks, GAN-C, a unified training

The associate editor coordinating the review of this manuscript and approving it for publication was Inês Domingues.

framework for GANs to reach Nash equilibrium effectively. Our GAN-C introduces a novel loss function into the standard GANs and works with a constraint that explicitly controls the output of the discriminator.

Although the original loss function applied in standard GANs and the proposed one ideally end up with the same Nash equilibrium, the latter does show a better convergence and score higher in our experiments. Our contributions can be summarized as:

- We propose Constrained Generative Adversarial Networks, GAN-C, that introduces a constraint into the loss function so as to closely reach to the Nash equilibrium for training and obtaining better generative data.
- Unlike the standard GANs and many of its variants, our proposed GAN-C explicitly controls the discriminator's output.
- We conduct a number of experiments to demonstrate the effectiveness of our proposed GAN-C [1] and provide an inspiration for future research on GANs: improving the training of GANs from the perspective of the discriminator's output.

The remainder of the paper is organized as: Section 2 discusses related work; Section 3 describes the proposed method; Section 4 describes our experimental setup; Section 5 is devoted to our experimental results; We conclude the paper in Section 6.

II. RELATED WORK

A. LOSS FUNCTIONS FOR TRAINING GANs AND ITS VARIANTS

Generative Adversarial Networks aim at training two networks, a generative network and an adversarial network, that compete against each other. The value function for the standard GANs [1] is defined as follows:

$$L(G, D) = \mathbb{E}_{\boldsymbol{x_r} \sim p_{data}(\boldsymbol{x_r})}[\log D(\boldsymbol{x_r})] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))], \quad (1)$$

where p_{data} and p_z denote the probability distribution of real data and the input prior noise z, and $\boldsymbol{x_r}$ denotes real data. We refer to (1) as a saturating version of the standard GANs.

For a fixed generator G, the optimal discriminator D^* in GANs is given by:

$$D^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + q_G(\boldsymbol{x})}, \quad (2)$$

where q_G denotes the generator $G(z)$'s distribution when $z \sim p_z(z)$. When D is optimal, optimizing G resembles minimizing the Jensen-Shannon divergence between the data and the model distribution: $JSD\left(p_{data} \| q_G\right)$. Meanwhile, the Jensen-Shannon divergence between two distributions is always non-negative and only equal to zero when two distributions are equal, i.e. $p_{data}(\boldsymbol{x}) = q_G(\boldsymbol{x})$.

In practice, the term $\log(1 - D(G(z)))$ in (1) will saturate in the early stage of learning when the generator is poor, and the discriminator can distinguish fake data from real data with high confidence. To solve this, a non-saturating version of the standard GANs [1] is given as follows:

$$L_D(\tilde{G}, D) = \mathbb{E}_{\boldsymbol{x_r} \sim p_{data}(\boldsymbol{x_r})}[\log D(\boldsymbol{x_r})] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))], \quad (3)$$

$$L_G(G, \tilde{D}) = -\mathbb{E}_{z \sim p_z(z)}[\log(D(G(z)))]. \quad (4)$$

This non-saturating loss ends up providing much stronger gradients early in learning.

Another objective function for training GANs is hinge loss, which is widely used in a number of the standard GANs' variants [19], [21]–[25], and is given by:

$$L_D = \mathbb{E}_{\boldsymbol{x_r} \sim p_{data}(\boldsymbol{x_r})}\left[min\big(0, -1 + f(\boldsymbol{x_r})\big)\right] + \mathbb{E}_{z \sim p_z(z)}\left[min\big(0, -1 - f(G(z))\big)\right], \quad (5)$$

$$L_G = -\mathbb{E}_{z \sim p_z(z)}\left[f\big(G(z)\big)\right] \quad (6)$$

for the discriminator D and the generator G respectively, where $D(\boldsymbol{x}) = \sigma(f(\boldsymbol{x})) = \frac{1}{1+\mathrm{e}^{-f(\boldsymbol{x})}}$. Optimizing hinge loss is equivalent to minimizing the reverse Kullback-Leibler(KL) divergence: $KL(q_G \| p_{data})$.

B. NASH EQUILIBRIUM

The basic idea behind GANs is to build a game between the discriminator and the generator for them to compete against each other. In this game, the discriminator learns to discriminate between real and fake data while the generator learns to deceive the discriminator. The solution to this game is called Nash equilibrium [26]. If both models are given enough capacity, the Nash equilibrium of this game is achieved when $p_{data}(\boldsymbol{x}) = q_G(\boldsymbol{x})$, and for all $\boldsymbol{x}$, $D^*(\boldsymbol{x}) = 0.5$.

However, in practice, it is often observed that GANs can be hard to converge. In recent years, a lot of research has been targeted at solving this problem and one of them has indicated that if, at the point where equilibrium is achieved, the eigenvalues of the Jacobian only falls into the negative real-part, the training of GAN can converge locally with a small learning rate [27], [28]. It has been proved that the optimal solution to (1) with $p_{data}(\boldsymbol{x}) = q_G(\boldsymbol{x})$ and $D^*(\boldsymbol{x}) = 0.5$ is actually a unique Nash equilibrium of the game [15]. So theoretically, making sure that the discriminator converging to 0.5 is key to finding the Nash equilibrium in GANs.

III. THE PROPOSED METHOD

A. MODIFIED LOSS FUNCTION WITH CONSTRAINT

To improve the training of GANs, some researchers tend to use loss functions based on reverse KL divergence because models trained with this divergence would prefer to generate samples from certain modes in the training distribution while ignoring others. However, it may lead to poor performance due to its asymmetry.

Ideally, when training GANs to find Nash equilibrium by optimizing (1), $D(\boldsymbol{x})$ will converge to 0.5. This is only

[1] The source code of our GAN-C is publicly available from: https://github.com/cxp504/Constrained_GAN

achieved when assuming that the discriminator is always trained to optimality during every step of alternate training between the discriminator and generator. Let $\mathcal{S}$ denote the training dataset that contains real data. In the k-th iteration, the discriminator is optimized to:

$$D^k(\boldsymbol{x}) = \frac{p^k_{data}(\boldsymbol{x})}{p^k_{data}(\boldsymbol{x}) + q^k_G(\boldsymbol{x})}.$$

Then we optimize the generator to make the generator's distribution further approach the real data distribution. Later in the $(k+1)$-th iteration, the discriminator is further optimized to:

$$D^{k+1}(\boldsymbol{x}) = \frac{p^{k+1}_{data}(\boldsymbol{x})}{p^{k+1}_{data}(\boldsymbol{x}) + q^{k+1}_G(\boldsymbol{x})}.$$

When $\boldsymbol{x} \in \mathcal{S}, p^k_{data}(\boldsymbol{x})$ is equal to $p^{k+1}_{data}(\boldsymbol{x})$, and $q^k_G(\boldsymbol{x}) \leq q^{k+1}_G(\boldsymbol{x})$, representing that the generator's distribution is approaching the real data distribution. Therefore, we have $D^k(\boldsymbol{x}) \geq D^{k+1}(\boldsymbol{x})$. Additionally, during training, the generator is learning to deceive the discriminator and as a result, $D\big(G(z)\big)$ will gradually increase as the generated samples become harder to distinguish. Note that here we are referring to the whole trend, and this may not specifically apply to every single $\boldsymbol{x}$ or z. To conclude, as training goes on, the discriminator's output for real data will ideally decrease and on the other side, increase for generated samples.

Unfortunately, without any stabilization factor, optimizing (1) directly may prevent the discriminator from convergence and unexpectedly lead to exploding gradient problem if the non-saturating version is used. This is because in practice, when optimizing the generator, the discriminator's output for generated samples will increase but remain unchanged for real data, while optimizing the discriminator results in an increase in its output for real data and correspondingly a decrease for generated samples. Moreover, the generator performs relatively poor early in training and if the support of the generator's distribution and the support of the real data distribution are disjoint, there exists a discriminator that can perfectly distinguish between real and fake samples [17].

In this paper, it is argued that this is due to the lack of an explicit constraint on the discriminator's relative output about real samples and fake samples when optimizing (1). Given that the discriminator will converge to 0.5 when Nash equilibrium is achieved, a constraint as follows is adopted to explicitly help the discriminator converge:

$$\begin{aligned} h(G, D) = \mathbb{E}_{\{x_r \sim p_{data}(x_r), z \sim p_z(z)\}} \big[&\log D(\boldsymbol{x_r}) \\ &- \log D\big(G(z)\big)\big]^2 \leq \varepsilon, \end{aligned} \tag{7}$$

where ε represents the biggest constant that guarantees stable training and does not depend on the discriminator's parameter. When $D(\boldsymbol{x_r}) = D(G(z)), h(G, D) = 0$ and when $|D(\boldsymbol{x_r}) - D(G(z))| = 1, h(G, D) = +\infty$. To some extent, $h(G, D)$ can be regarded as the degree to which the discriminator's output diverges.

Combined with (1), the proposed objective function is given as follows:

$$\begin{aligned} L^{new}_D(\tilde{G}, D) &= L_D(\tilde{G}, D) - \lambda h(\tilde{G}, D) \\ &= \mathbb{E}_{\boldsymbol{x_r} \sim p_{data}(\boldsymbol{x_r})} \big[\log D(\boldsymbol{x_r})\big] + \mathbb{E}_{z \sim p_z(z)} \\ &\quad \times \big[\log\big(1 - D(G(z))\big)\big] - \lambda h(\tilde{G}, D) \end{aligned} \tag{8}$$

$$\begin{aligned} L^{new}_G(G, \tilde{D}) &= \mathbb{E}_{\boldsymbol{x_r} \sim p_{data}(\boldsymbol{x_r})} \big[\log D(\boldsymbol{x_r})\big] \\ &\quad \mathbb{E}_{z \sim p_z(z)} \big[\log\big(1 - D(G(z))\big)\big], \end{aligned} \tag{9}$$

where $\lambda \in \mathbb{R}^+$, and $h(\tilde{G}, D)$ restricts the discriminator's output to prevent it from severe divergence and causing instability in training. Algorithm 1 shows how to train GANs with our method. Particularly, this objective becomes the non-saturating standard loss when $\lambda = 0$. If the difference between $D(\boldsymbol{x_r})$ and $D(G(z))$ becomes too large during training, maximizing $L^{new}_D(\tilde{G}, D)$ will cause the discriminator to lie in the function space where $h(G, D)$ decreases to make sure that the training proceeds steadily.

Algorithm 1 GAN With Constraint h. We Use Default Value of $\lambda = 0.3$

```
Require: The number of discriminator iterations $n$, the batch size $N$.
initialize the discriminator parameters $\theta_D$ and initialize the generator parameters $\theta_G$.
for number of training iterations do
    for n steps do
        Sample noise data $\{z_i\}_{i=1}^N \sim p_z$
        and real data $\{x_i\}_{i=1}^N \sim p_{data}$;
        Update the discriminator parameters:
        $\theta_D \leftarrow Adam\left(\nabla_{\theta_D} \frac{1}{N} \sum_{i=1}^N L^{new}_D(\tilde{G}, D)\right)$.
    end for
    Sample noise data $\{z_i\}_{i=1}^N \sim p_z$;
    Update the generator parameters:
    $\theta_G \leftarrow Adam\left(\nabla_{\theta_G} \frac{1}{N} \sum_{i=1}^N L^{new}_D(G, \tilde{D})\right)$.
end for
```

In this way, we transfer the minimax optimization problem that GANs aim to solve from the problem (i), i.e., $\min_G \max_D L(G, D)$ to the problem (ii), i.e., $\max_D L^{new}_D(G, D)$, $\min_G L^{new}_G(G, D)$. Obviously, we can see that $p_{data}(\boldsymbol{x}) = q_G(\boldsymbol{x})$ and $D^*(\boldsymbol{x}) = 0.5$ are the optimal solution to both problem (i) and (ii).

On the other side, as GANs are highly nonconvex in practice, there could be many local Nash Equilibria. Furthermore, when the discriminator and generator have limited capacity, optimal Nash equilibrium does not necessarily exist. Nevertheless, our experiments still show that our method yeilds better results with local Nash equilibria. In addition, it should be noted that our method does not take full advantage of the adversarial training when $\lambda \neq 0$. The larger λ is, the stronger the convergence ability of the discriminator's output is. If λ approaches infinity, then the network will not take the advantage of the adversarial training. λ controls the tradeoff between the adversarial training and the discriminator's output. Although our method sacrifices a bit of adversarial training, it makes the generator find a better generative distribution during the training process. Our experiments in Section V show that our method can generate better quality data in most cases than the methods that take full advantage of the adversarial training.

B. RELATION AND DIFFERENCE BETWEEN CONSTRAINT h AND LIPSCHITZ CONSTRAINT

Enforcing Lipchitz constraint on the discriminator is an important method to stabilize training and also to indirectly restrict the discriminator's output. Furthermore, using this and our method simultaneously exerts a better constraint on the discriminator's output.

The form of K-Lipschitz continuous functions can be defined as:

$$Lip = \frac{\left\| f(\boldsymbol{x_r}) - f(\boldsymbol{x_f}) \right\|_2^2}{\left\| \boldsymbol{x_r} - \boldsymbol{x_f} \right\|_2^2} \leq K, \tag{10}$$

where $\boldsymbol{x_f} = G(z)$ represents the generated sample, and K is a constant. In practice, the input data of the discriminator should be normalized so that $\|\boldsymbol{x_r} - \boldsymbol{x_f}\|_2^2$ is bounded. Then constraining $\left\|f(\boldsymbol{x_r}) - f(\boldsymbol{x_f})\right\|_2^2$ is equivalent to constraining Lip. Also, the relation between $\left[\log D(\boldsymbol{x_r}) - \log D(\boldsymbol{x_f})\right]^2$ and $\left\|f(\boldsymbol{x_r}) - f(\boldsymbol{x_f})\right\|_2^2$ is as follows:

$$\begin{aligned}
&\Big[\log D(\boldsymbol{x_r}) - \log D\big(G(z)\big)\Big]^2 \\
&\quad = \Big\| \log D(\boldsymbol{x_r}) - \log D(\boldsymbol{x_f}) \Big\|_2^2 \\
&\quad = \left\| \log \frac{e^{f(\boldsymbol{x_r})}}{1 + e^{f(\boldsymbol{x_r})}} - \log \frac{e^{f(\boldsymbol{x_f})}}{1 + e^{f(\boldsymbol{x_f})}} \right\|_2^2 \\
&\quad = \left\| f(\boldsymbol{x_r}) - f(\boldsymbol{x_f}) + \log \frac{1 + e^{f(\boldsymbol{x_f})}}{1 + e^{f(\boldsymbol{x_r})}} \right\|_2^2 \\
&\quad \leq \left\| f(\boldsymbol{x_r}) - f(\boldsymbol{x_f}) + \log \left(1 + \frac{e^{f(\boldsymbol{x_f})}}{1 + e^{f(\boldsymbol{x_r})}}\right) \right\|_2^2 \\
&\quad \leq \left\| f(\boldsymbol{x_r}) - f(\boldsymbol{x_f}) \right\|_2^2 + \left\| \log\left(1 + e^{f(\boldsymbol{x_f}) - f(\boldsymbol{x_r})}\right) \right\|_2^2 \\
&\quad \leq \left\| f(\boldsymbol{x_r}) - f(\boldsymbol{x_f}) \right\|_2^2 + \left\| \left|f(\boldsymbol{x_f}) - f(\boldsymbol{x_r})\right| + \log 2 \right\|_2^2 \\
&\quad \leq 2 \left\| f(\boldsymbol{x_r}) - f(\boldsymbol{x_f}) \right\|_2^2 + \log^2 2,
\end{aligned} \tag{11}$$

and

$$\begin{aligned}
&\Big[\log D(\boldsymbol{x_r}) - \log D\big(G(z)\big)\Big]^2 \\
&\quad = \Big\| \log D(\boldsymbol{x_r}) - \log D(\boldsymbol{x_f}) \Big\|_2^2 \\
&\quad = \left\| \log \frac{e^{f(\boldsymbol{x_r})}}{1 + e^{f(\boldsymbol{x_r})}} - \log \frac{e^{f(\boldsymbol{x_f})}}{1 + e^{f(\boldsymbol{x_f})}} \right\|_2^2 \\
&\quad = \left\| f(\boldsymbol{x_r}) - f(\boldsymbol{x_f}) + \log \frac{1 + e^{f(\boldsymbol{x_f})}}{1 + e^{f(\boldsymbol{x_r})}} \right\|_2^2 \\
&\quad = \left[f(\boldsymbol{x_r}) - f(\boldsymbol{x_f})\right]^2 + \left[\log \frac{1 + e^{f(\boldsymbol{x_f})}}{1 + e^{f(\boldsymbol{x_r})}}\right]^2 \\
&\quad \geq \left\| f(\boldsymbol{x_r}) - f(\boldsymbol{x_f}) \right\|_2^2.
\end{aligned} \tag{12}$$

Therefore, combining (10), (11), and (12) we have:

$$\begin{aligned}
&\mathbb{E}_{\{\boldsymbol{x_r} \sim p_{data}(\boldsymbol{x_r}), \boldsymbol{x_f} \sim q_G(\boldsymbol{x_f})\}} \\
&\quad \times \left[\left\|\boldsymbol{x_r} - \boldsymbol{x_f}\right\|_2^2 Lip\right] \leq h(G, D) \\
&\quad \leq \mathbb{E}_{\{\boldsymbol{x_r} \sim p_{data}(\boldsymbol{x_r}), \boldsymbol{x_f} \sim q_G(\boldsymbol{x_f})\}} \\
&\quad \times \left[\left\|\boldsymbol{x_r} - \boldsymbol{x_f}\right\|_2^2 (2Lip + \log^2 2)\right]
\end{aligned} \tag{13}$$

Obviously, if Lipschitz constraint is enforced on the discriminator, the variation of $D(\boldsymbol{x_r})$ and $D(\boldsymbol{x_f})$ will be indirectly restricted and likewise, constraining h, to some extent, restricts the function space that the discriminator lies within. However, our proposed constraint h and Lipschitz constraint differ slightly in terms of restricting the discriminator function. Ours allows part of the combined samples $(\boldsymbol{x_r}, \boldsymbol{x_f})$ not to satisfy (10) while Lipschitz constraint requires that (10) is satisfied for any $(\boldsymbol{x_r}, \boldsymbol{x_f})$. Despite being able to stabilize training, Lipschitz constraint actually shrinks the function space from which the discriminator can choose.

In theory, combining these two constraints together will end up giving better performance, with Lipschitz constraint playing a part in stabilizing and ours pushing the discriminator to choose those that satisfy $D(\boldsymbol{x}) = 0.5$. In practice, after a certain number of iterations, the networks will come to a state of dynamic balance where follow-up updates may not significantly boost performance and the discriminator begins to meander in a small area of the function space. At this time, our constraint will provide better guidance for the discriminator to choose the function that gives the best convergence, thus preventing the discriminator from severe deviation from Nash equilibrium. In the next section, we validate the efficacy of our method and show that combining our constraint with other loss functions rather than the standard one can also improve the quality of generated samples.

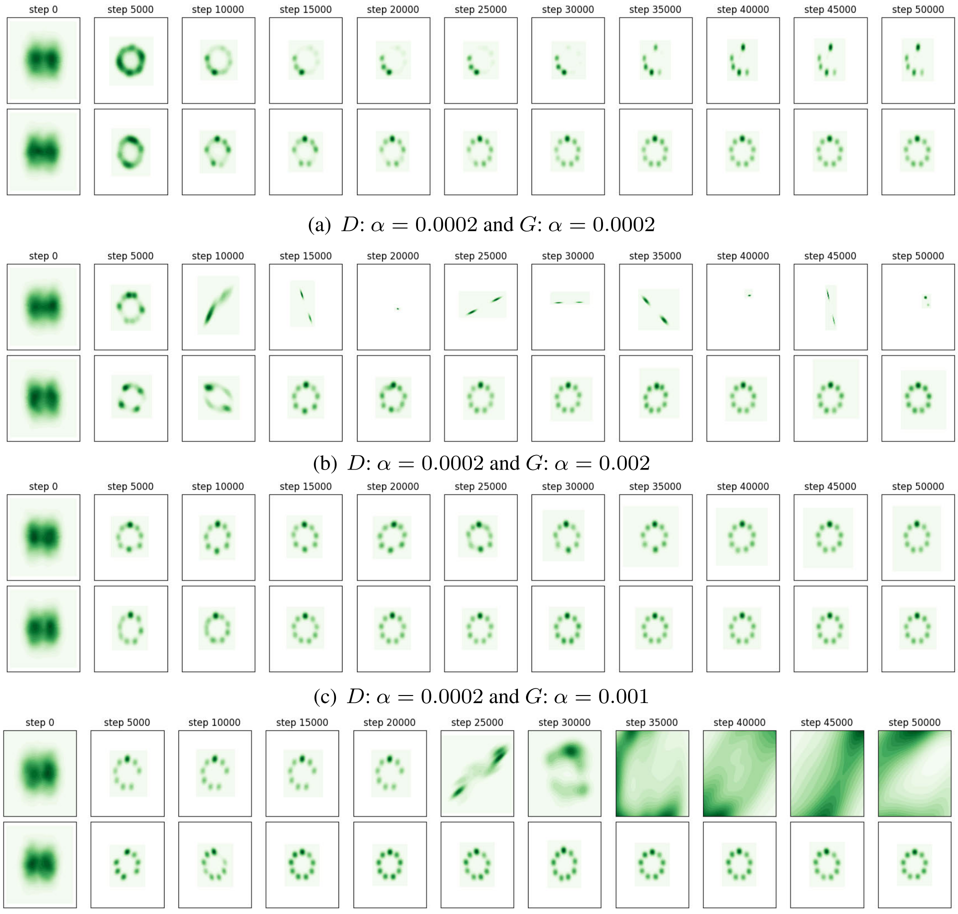
C. RELATION TO RELATIVISTIC GANs

The non-saturating loss function of Relativistic standard GANs (RGAN) is given as:

$$\begin{aligned}
L_D = &-\mathbb{E}_{\{\boldsymbol{x_r} \sim p_{data}(\boldsymbol{x_r}), z \sim p_z(z)\}} \\
&\times \left[\log(sigmoid(f(\boldsymbol{x_r}) - f\big(G(z)\big)))\right],
\end{aligned} \tag{14}$$

$$\begin{aligned}
L_G = &-\mathbb{E}_{\{\boldsymbol{x_r} \sim p_{data}(\boldsymbol{x_r}), z \sim p_z(z)\}} \\
&\times \left[\log(sigmoid(f\big(G(z)\big) - f(\boldsymbol{x_r})))\right].
\end{aligned} \tag{15}$$

When optimizing the generator, RGAN guarantees that $f(\boldsymbol{x_r})$ decreases as $f(G(z))$ increases, and thus helps the discriminator to converge. However, our method focuses on helping the discriminator to converge in a more direct way while



(a) D: $\alpha = 0.0002$ and G: $\alpha = 0.0002$

(b) D: $\alpha = 0.0002$ and G: $\alpha = 0.002$

(c) D: $\alpha = 0.0002$ and G: $\alpha = 0.001$

(d) D: $\alpha = 0.001$ and G: $\alpha = 0.001$

FIGURE 1. Comparison of generation performance over iterations between the 2-dimensional data generated by the standard generative adversarial network, i.e., Standard GAN, and our proposed model, Constrained GAN. The training data are the 2-dimensional mixture data from nine Gaussians, which are distributed in circles. We did the generation experiments four times, resulting in the comparison shown in (a), (b), (c) and (d), with the figures at the top line and the bottom line showing the 2-dimensional distributions of the generated data from the Standard GAN and our Constrained GAN, respectively. The parameters α for the corresponding discriminators *D* and the generators *G* are provided in the captions of the figures.

optimizing the discriminator. In the later experiments, we validate that our method achieves better FID scores compared to RGAN and is able to generate images with higher quality.

IV. EXPERIMENTAL SETUP

To demonstrate our method, we conduct several unsupervised image generation experiments on mixture of 9 Gaussians [29], MNIST [30], CIFAR-10 [31], STL-10 [32], FFHQ [33], and CAT [34]. In this section, we first detail the dataset used throughout our experiments, and then we train GAN to generate a 2D mixture of 9 Gaussians and MNIST images. Next, we evaluate the performance of the network model on CIFAR-10 and STL-10 using different loss functions, and more complex FFHQ and CAT datasets with higher resolution. It is worth noting that our proposed method is inspired by Nash equilibrium and is aimed at preventing the discriminator's output from deviating from Nash equilibrium. From the perspective of the discriminator's output, our method is more reasonable. We argue that this deviation affects the quality of generated samples and also show through experiments that our method can improve such deviation. Meanwhile, given that experiments done in previous research are all based on the non-saturating version of (1), our experiments are also based on the non-saturating version for the sake of fairness.

A. DATASETS

We use several public datasets, and the datasets are described as follows:

- *MNIST:* This dataset [30] contains a training set with $60,000$ images of digits and a test set with $10,000$ images. Each image is a 28×28 greyscale image.
- *CIFAR-10:* This dataset [31] contains $60,000$ 32×32 color images in 10 classes, each of which consists of $6,000$ images. There are $50,000$ training images and $10,000$ training images.

- *STL-10:* This dataset [32] is a dataset containing 13, 000 labeled images and 100, 000 unlabeled images in 10 classes. Each is a 96 × 96 color image. In our experiments, we choose 100000 unlabeled images as our training set with each image scaled to a 48 × 48 color image.
- *FFHQ:* This dataset [33] consists of 70, 000 in-the-wild face images at 128 × 128 and 1, 024 × 1, 024 resolutions. Due to computational constraints, we select the first 10, 000 images as the training set at the 128 × 128 resolution.
- *CAT:* This dataset [34] contains ∼ 10, 000 images with annotations. We preprocess the dataset the same way as what we did for RGAN, including cropping the images to the faces of the cats and removing some inappropriate images. As a result, our training set contains 9,071 images and each of them is scaled to 64 × 64, 128 × 128, and 256 × 256, respectively.

V. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we report and analyze our experimental results on Gaussian datasets and the image datasets.

A. *GAUSSIAN DATA GENERATION*

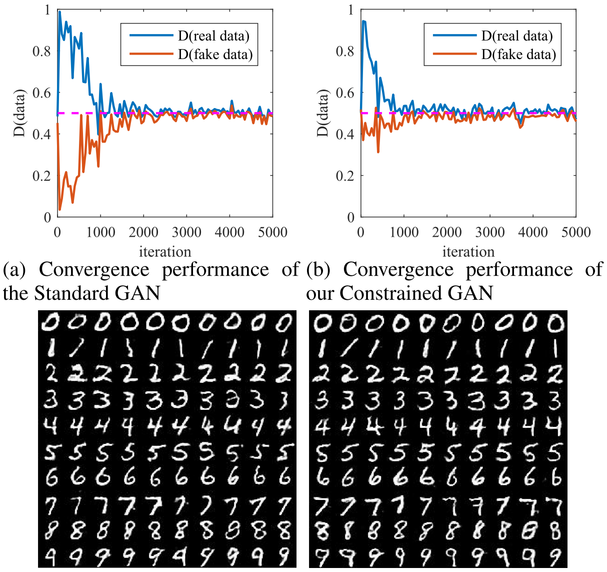
In this section, we aim at generating 2-dimensional data by training the 2-dimensional Gaussian data (data distributed in circles) obtained from nine 2-dimensional Gaussians, and make the generation performance comparisons between our proposed model, i.e., Constrained GAN, and the baseline model, i.e., the standard GAN [29].

1) IMPLEMENTATION DETAILS

To illustrate the generation effectiveness of our proposed method, we make the generation performance comparison between the 2-dimensional data generated by our proposed Constrained GAN, and those generated by the standard generative adversarial model, Standard GAN. We implement both our model and the baseline model by fully connected networks proposed in [29], and train both of the models by the same data obtained from nine 2-dimensional Gaussians. ADAM [35] optimizer is used throughout all experiments with the momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$ and different learning rates α [36]. In addition, the number of iterations for updating the generator is set to 50k. For convenient discussion, we refer the standard generative adversarial network proposed in [1] as Standard GAN and the network proposed in this paper as Constrained GAN. Hyper-parameter λ is set to 0.3 unless specially noted.

2) GENERATION PERFORMANCE

The distributions of the 2-dimensional data generated by our Constrained GAN, and the Standard GAN, over iterations are shown in Figure 1. The figures at the top line and the bottom in Figure 1 (a), (b), (c) and (d) show the distributions of the generated data from the Standard GAN and our model, Constrained GAN, respectively. We have two findings according to Figure 1: (1) Our Constrained GAN model is able to generate the 2-dimensional Gaussian data distributed as shown in a circle faster than Standard GAN model; (2) In the figures, our Constrained GAN model is always able to generate 2-dimensional Gaussian data with iterations evolve, while Standard GAN fails to generate such data in some cases such as those shown in Figure 1 (b) and (d). These findings confirms the merits of our Constrained GAN model that it is robust to generate data needed.



(a) Convergence performance of the Standard GAN
(b) Convergence performance of our Constrained GAN
(c) Samples generated by the Standard GAN
(d) Samples generated by our Constrained GAN

FIGURE 2. Convergence performance of the discriminator's output of (a) the Standard GAN and (b) the Constrained GAN, and samples generated by the generators trained by different methods: (c) the Standard GAN and (d) the Constrained GAN.

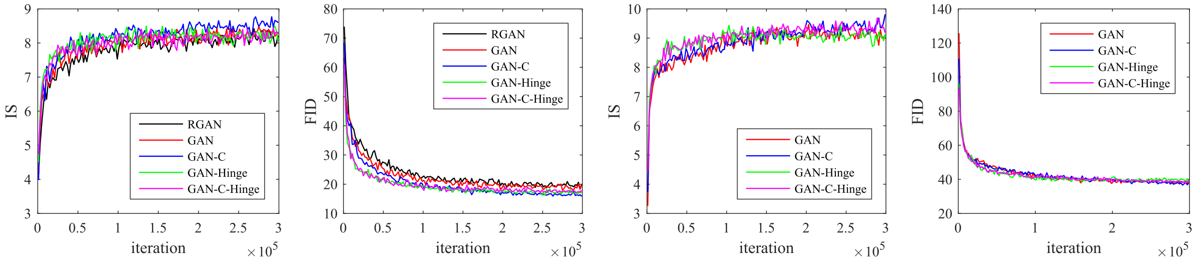
B. *MNIST IMAGE GENERATION*

In this section, we detail the implementation of our model and the baselines on MNIST dataset, and report the experimental results.

1) IMPLEMENTATION DETAILS AND RESULTS

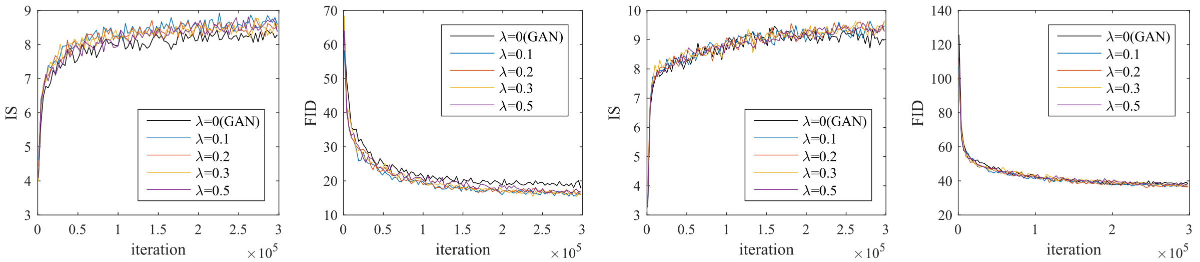
We test on MNIST dataset using the Convolutional Neural Network architecture proposed in [37]. ADAM optimizer [35] is used throughout all experiments with the momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$ and the learning rate $\alpha = 0.0002$ [36]. In addition, the update ratio of discriminator to generator is set to 1 : 1, and the number of updates for the generator is set to 5 k. We conduct a comparative experiment between our Constrained GAN and the standard GAN with batch norm [38] applied to the generator and spectral normalization applied to the discriminator.

The curves of the discriminator's output and samples generated by the generators trained with different methods are shown in order in Figure 2. As shown in the figure, our Constrained GAN converges faster than the Standard GAN for the output of the corresponding discriminators, and both methods can generate high-quality samples though.



(a) IS on CIFAR-10 (b) FID on CIFAR-10 (c) IS on STL-10 (d) FID on STL-10

FIGURE 3. Inception scores and FIDs with respect to different loss functions on ResNet.



(a) IS on CIFAR-10 (b) FID on CIFAR-10 (c) IS on STL-10 (d) FID on STL-10

FIGURE 4. The curves of IS and FID based on ResNet with different λ (0,0.1,0.2,0.3,0.5).

C. CIFAR-10 AND STL-10 IMAGE GENERATION

In this section, we detail the evaluation metrics used to evaluate the performance of our model and the baselines on CIFAR-10 and STL-10 datasets, the implementation, and the comparison analysis.

1) EVALUATION METRICS

In our experiments, two metrics: Inception Score (IS) [14] and Fréchet inception distance (FID) [16], are adopted to serve as quantitative measures. Inception Score (IS) is defined as:

$$I(\{\boldsymbol{x_i}\}_{i=1}^{N}) := \exp(\mathbb{E}\big[D_{KL}[p(y|\boldsymbol{x}) \parallel p(y)]\big]),$$

where $p(y)$ is approximated by $\frac{1}{N}\sum_{i=1}^{N} p(y|\boldsymbol{x_i})$ and $p(y|\boldsymbol{x})$ is estimated by a pretrained Inception Net [40]. The Inception score computes the KL divergence between distributions $p(y|\boldsymbol{x})$ and $p(y)$. Higher IS means better generative quality. Fréchet inception distance (FID) uses the 2nd order information of the final layer of the inception model applied to the examples. Initially, Fréchet distance (FD) [41] is 2-Wasserstein distance between two Gaussian distribution p_1 and p_2:

$$FD := \|\boldsymbol{\mu_{p_1}} - \boldsymbol{\mu_{p_2}}\|_2^2 + \mathrm{tr}(\textstyle\sum_{p_1} + \sum_{p_2} - 2(\sum_{p_1}\sum_{p_2})^{1/2}),$$

where $\{\boldsymbol{\mu_{p_1}}, \sum_{p_1}\}$, $\{\boldsymbol{\mu_{p_2}}, \sum_{p_2}\}$ denotes the mean and covariance of p_1 and p_2 respectively. So FID between two image distribution p_1 and p_2 is the FD between $f_{incept}(p_1)$ and $f_{incept}(p_2)$, i.e. the distribution after the inception net transformation. Lower FID means better generative quality as well as diversity.

2) IMPLEMENTATION DETAILS

To further validate the efficacy of our method, the compared experiment are conducted on CIFAR-10 and STL-10. All experiments are based on Chainer framework and built with either CNN architecture or ResNet architecture as described in [19]. As for optimizer, ADAM [35] is used for all experiments but the setting of hyper-parameter differs with different objective functions. For (1) or our novel standard loss, the hyper-parameters are set as follows: $\alpha = 0.0002$, $\beta_1 = 0.5$, $\beta_2 = 0.999$ [36]. For hinge loss, we have $\alpha = 0.0002$, $\beta_1 = 0.$, $\beta_2 = 0.9$ [19] on ResNet. The update ratio of discriminator to generator is set to $1 : 1$ if CNN is used and $5 : 1$ if ResNet is used. Also, spectral normalization is applied to the discriminator to stabilize training. At every 1000 iterations, 5000 samples generated by the generator will be used for evaluation. Table 1 shows the results with the number of iterations at 100k and all experiments are repeated 10 times with random initialization on CIFAR-10. We use the same ResNet architecture as described in [19]. Table 2 shows the results with the number of iterations at 300 k and all experiments are repeated 3 times with random initialization. Note that in the table, GAN represents optimizing (3)(4), GAN-Hinge represents optimizing (5)(6), GAN-C represents optimizing our proposed loss function and GAN-C-Hinge represents optimizing (5)(6) with our proposed constraint.

3) RESULTS ON CIFAR-10 AND STL-10

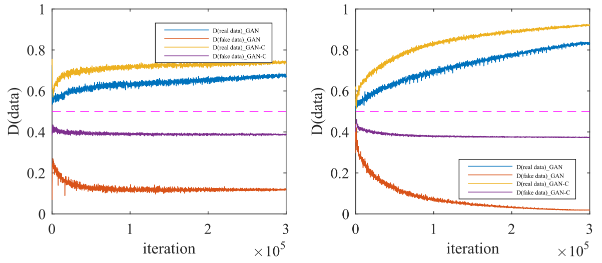
In Table 1, we can see that, compared to other methods, our GAN-C performs better with ResNet on CIFAR-10. We also increase the number of iterations to 300 k and as the results shown in Table 2, we can see that compared with

TABLE 1. Inception scores on CIFAR-10 using different methods without label conditioning(UNSUPERVISED) and with label conditioning(SUPERVISED).

Method	UNSUPERVISED Score	SUPERVISED Score
Real Data	11.24±.12	11.24±.12
DCGAN [36]	6.16±.07	6.58
Improved GAN [14]	6.86±.06	8.09±.07
WGAN-GP [18]	7.86±.07	8.42±.10
SplittingGAN [39]	7.90±.09	8.87±.09
SNGAN [19]	8.22±.05	8.60±.08
RGAN	8.15±.14	–
GAN-C(Ours)	**8.53±.09**	**9.00±.08**

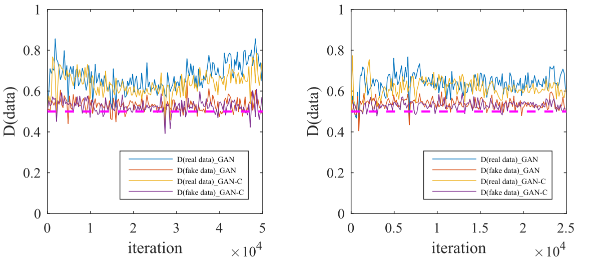
TABLE 2. Inception scores and FIDs on CIFAR-10 and STL-10 at 300k iterations using different loss functions without label conditioning.

Loss	Inception Score		FID	
	CIFAR-10	STL-10	CIFAR-10	STL-10
Real Data	11.24±.12	26.08±.26	7.8	7.9
-CNN-				
RGAN	7.30±.04	–	29.02±.29	–
GAN	7.64±.04	8.49±.05	26.72±.52	44.31±.46
GAN-Hinge	7.72±.06	8.77±.06	24.83±.30	43.07±.42
GAN-C-Hinge(Ours)	**7.78±.03**	**8.95±.03**	**24.75±.47**	**41.67±.56**
GAN-C(Ours)	7.76±.07	8.74±.10	25.27±.10	42.65±.43
-ResNet-				
RGAN	8.47±.04	–	18.51±.14	–
GAN	8.57±.06	9.42±.06	17.95±.11	38.20±.24
GAN-Hinge	8.56±.05	9.45±.05	16.80±.29	38.22±.67
GAN-C-Hinge(Ours)	8.73±.04	9.76±.08	15.40±.26	36.57±.15
GAN-C(Ours)	**8.96±.07**	**9.77±.06**	**15.28±.08**	**36.05±.20**



(a) The discriminator's output on CIFAR-10 (b) The discriminator's output on STL-10

FIGURE 5. The discriminator's output based on ResNet.



(a) The discriminator's output on FFHQ (b) The discriminator's output on CAT

FIGURE 6. The discriminator's output based on different datasets.

GAN, GAN-C achieves better scores on datasets. GAN-C-Hinge does not seem to overperform GAN-Hinge when using CNN. One possible reason is that the network architecture is not complicated enough to have sufficient capacity given that our proposed method does perform obviously better than any others when using ResNet. This also indicates that given sufficient capacity, optimizing our proposed loss function will score higher. In fact, even at 200Kth iteration, our method has already shown a great advantage over the others on ResNet. Meanwhile, it is also shown that our method boosts the performance of hinge loss and yeilds samples with high-quality.

4) COMPARISON

We also record the Inception scores and FIDs of different loss functions based on ResNet (see Figure 3). We can see that the scores of GAN and GAN-Hinge tend to plateau around 20K-th iterations, while GAN-C keeps improving even afterward. Moreover, to study the influence of the hyperparameter λ to the quality of the generated samples, we also show the curves of Inception scores and FIDs with different λ on ResNet. As shown in Figure 4, our method is robust with respect to the change of λ. In Figure 5, we plot the curves of the discriminator's output based on ResNet, from which we can see that the constraint we adopt actually affects the discriminator's output. When the output plateaus and the network cannot find the Nash equilibrium, GAN-C can find a better generation distribution to generate higher quality images.

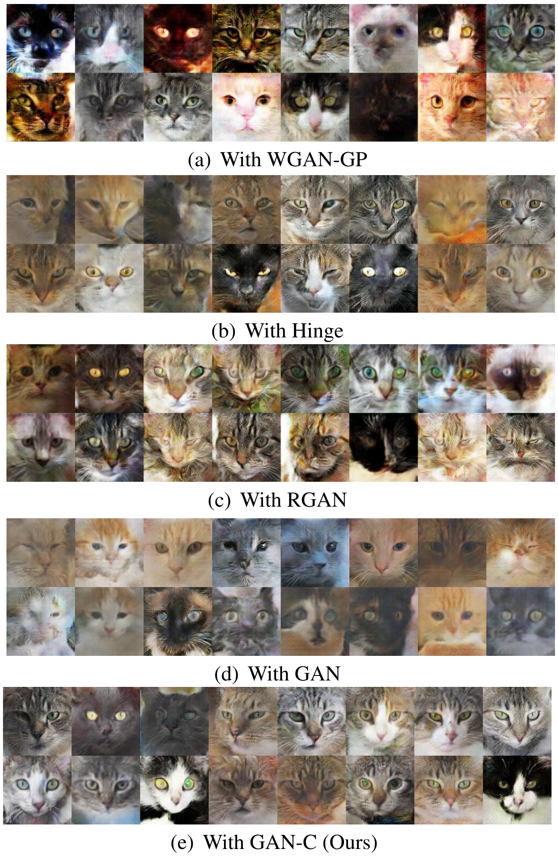
D. FFHQ AND CAT IMAGE GENERATION

In this section, we validate the efficacy of our method on FFHQ and CAT. These two datasets are more complex and higher resolution datasets, bringing more challenges to the generation of images.

1) IMPLEMENTATION DETAILS

We train with different loss functions on CAT with different resolutions and FFHQ. All experiments are conducted on Tensorflow framework and use the DCGAN [36] architecture. For WGAN-GP [18], the settings of its hyper-parameters are as: $\alpha = 0.0001, \beta_1 = 0., \beta_2 = 0.9, n = 5$. For RGAN, the settings of its hyper-parameters are as: $\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999, n = 1$. Also, we use batch norm [38] in both the corresponding discriminator and the generator. For Hinge, GAN, and GAN-C (Ours), we set $\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999, n = 1$ and use spectral norm in the discriminator and batch norm in the generator.

We randomly select 8,000 real images from the training set and 4,000 samples generated by the generator for FID evaluation on 64 × 64 and 128 × 128. Moreover, due to the computational resource limit, we randomly select 2,000 real images from the training set and 2,000 samples generated by the generator in 256 × 256 resolution. Table 3 shows the results with the number of iterations at 50k. Table 4 shows the results with the number of iterations at 25k and all



(a) With WGAN-GP

(b) With Hinge

(c) With RGAN

(d) With GAN

(e) With GAN-C (Ours)

FIGURE 7. **128 × 128 cats with different methods: (a) with WGAN-GP; (b) with Hinge; (c) with RGAN; (d) with GAN; and (e) with GAN-C.**

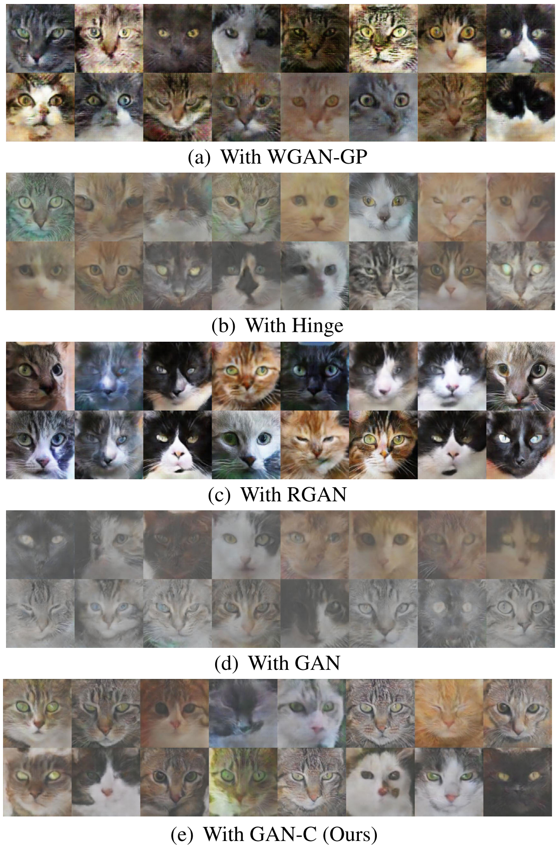
TABLE 3. **Minimum (MIN), Mean, and standard deviation (SD) of the FID on FFHQ using different methods. Lower FID indicates better generative quality. GAN-C(BN) denotes the GAN model using batch norm in the discriminator and the generator.**

Method	MIN	Mean	SD
WGAN-GP	123.03	128.97	4.25
RGAN	63.20	73.39	7.85
GAN	56.00	61.61	4.12
GAN-C	56.08	58.80	**1.96**
GAN-C(BN)	**50.94**	**54.34**	2.64

experiments are repeated 3 times with random initialization. Note that in the table, a missing number indicates the method do not converge in our experiments.

2) RESULTS ON FFHQ AND CAT

From Table 3, we can see that our method obtain lowest mean value and standard deviation of FID on FFHQ, which demonstrates that our method can generate more stable images. From Table 4, we can see that our GAN-C outperforms all other methods on CAT, achieving the lowest FID scores. Compared to both RGAN and Hinge, our method shows significant improvements, indicating that the constraint we introduce does help improve generative quality on 64 × 64 and 128×128. In 256×256 resolution, both Hinge and GAN-C have close mean of FID, but GAN-C actually achieves lower minimum FID. In Table 5, we change the



(a) With WGAN-GP

(b) With Hinge

(c) With RGAN

(d) With GAN

(e) With GAN-C (Ours)

FIGURE 8. **256 × 256 cats with different methods: (a) with WGAN-GP; (b) with Hinge; (c) with RGAN; (d) with GAN; and (e) with GAN-C.**

TABLE 4. **Minimum (min), mean, and standard deviation (SD) of the FID on CAT using different methods. Lower FID indicates better generative quality. GAN-C(BN) denotes the GAN model using batch norm in the discriminator and the generator.**

Method	64x64 images		128x128 images		256x256 images	
	min	mean±SD	min	mean±SD	min	mean±SD
WGAN-GP	36.59	39.22±2.55	38.10	41.59±2.90	58.71	64.78±4.43
Hinge	27.00	32.99±4.28	16.98	17.93±1.13	20.17	21.40±1.05
RGAN	18.92	25.11±4.89	21.18	21.32±0.15	21.38	26.68±3.77
GAN	24.46	25.65±1.28	15.75	16.59±0.65	23.25	23.93±0.48
GAN-C	22.02	22.48±0.42	12.85	**13.65±0.60**	18.94	20.97±1.48
GAN-C(BN)	**16.51**	**18.26±1.26**	**12.05**	13.67±1.95	**18.69**	22.47±3.15

TABLE 5. **The FID performance on the CAT dataset using the different settings and batch norms in the discriminators and the generators. Setting A: $\alpha = 0.0001$ $\beta_1 = 0.5$, $\beta_2 = 0.999$ on 128 × 128 [42]. Setting B: $\alpha = 0.0001$ $\beta_1 = 0.5$, $\beta_2 = 0.9$ on 128 × 128 [18]. Setting C: $\alpha = 0.0002$ $\beta_1 = 0.5$, $\beta_2 = 0.9$ on 128 × 128. Setting D: $\alpha = 0.0001$ $\beta_1 = 0.5$, $\beta_2 = 0.999$ on 256 × 256.**

Method	A	B	C	D
WGAN-GP	28.96	26.96	20.83	62.55
Hinge	28.67	**15.98**	21.92	61.06
RGAN	35.78	33.93	29.23	–
GAN	27.84	16.65	26.60	60.85
GAN-C(Ours)	**26.63**	16.23	**17.23**	**58.30**

experimental settings, i.e., using different hyper-parameters and batch norms in discriminators and the generators, and make the FID performance comparisons. We observe that WGAN-GP has higher FID scores than our method. Also, GAN-C performs better than GAN and RGAN. The FID score of GAN-C is close to that of Hinge, albeit sightly worse with setting B. In addition to this, our method has the lowest FID scores. Overall, these results show that our method has lower FID scores with most settings than the methods that take full advantage of the adversarial training.

In Figure 6, we plot the curves of the discriminator's output on FFHQ and CAT, from which we can see that our method have the potential to help the discriminator converge slightly. Figure 7 and 8 show the samples generated by different methods on CAT.

VI. CONCLUSION

In this paper, we study the problem of closely and effectively reaching the Nash equilibrium during the training for the standard generative adversarial networks and their variants. To tackle this problem, we propose Constrained Generative Adversarial Networks, GAN-C, that introduces a constraint into the loss function. Our GAN-C is able to explicitly control the discriminator's output and obtain better generative data after training. In our experiments, we demonstrate that optimizing either our improved loss or the standard one will ideally give us the same solution while our proposed loss can help to stabilize the training process and obtain better generative data. Also, we validate the efficiency and potential of our method through experiments, and the experimental results show that our method performs better than the baseline models. As to future work, the constraint proposed in this paper may be of great value to investigate better mathematical expressions and other constraints may be proposed for training the standard generative adversarial networks and their variants. Moreover, the proposed method can take full advantage of the adversarial training on the premise that the generator can still generate high-quality data.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[2] Y. Zhao, S. Takaki, H.-T. Luong, J. Yamagishi, D. Saito, and N. Minematsu, "Wasserstein GAN and waveform loss-based acoustic model training for multi-speaker text-to-speech synthesis systems using a WaveNet vocoder," *IEEE Access*, vol. 6, pp. 60478–60488, 2018.

[3] H. M. Kasem, K.-W. Hung, and J. Jiang, "Spatial transformer generative adversarial network for robust image super-resolution," *IEEE Access*, vol. 7, pp. 182993–183009, 2019.

[4] O.-Y. Lee, Y.-H. Shin, and J.-O. Kim, "Multi-perspective discriminators-based generative adversarial network for image super resolution," *IEEE Access*, vol. 7, pp. 136496–136510, 2019.

[5] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2536–2544.

[6] L. Yuan, C. Ruan, H. Hu, and D. Chen, "Image inpainting based on patch-GANs," *IEEE Access*, vol. 7, pp. 46411–46421, 2019.

[7] Y. Jiang, J. Xu, B. Yang, J. Xu, and J. Zhu, "Image inpainting based on generative adversarial networks," *IEEE Access*, vol. 8, pp. 22884–22892, 2020.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2223–2232.

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1125–1134.

[10] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for Image-to-Image translation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2849–2857.

[11] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8789–8797.

[12] X. Li, C. Lin, R. Li, C. Wang, and F. Guerin, "Latent space factorisation and manipulation via matrix subspace projection," 2019, *arXiv:1907.12385*. [Online]. Available: http://arxiv.org/abs/1907.12385

[13] S. Liang, "Unsupervised semantic generative adversarial networks for expert retrieval," in *Proc. World Wide Web Conf.*, 2019, pp. 1039–1050.

[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.

[15] H. Ge, Y. Xia, X. Chen, R. Berry, and Y. Wu, "Fictitious GAN: Training GANs with historical models," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 119–134.

[16] 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 *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6626–6637.

[17] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.

[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[19] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," 2018, *arXiv:1802.05957*. [Online]. Available: http://arxiv.org/abs/1802.05957

[20] A. Jolicoeur-Martineau, "The relativistic discriminator: A key element missing from standard GAN," 2018, *arXiv:1807.00734*. [Online]. Available: http://arxiv.org/abs/1807.00734

[21] J. Hyun Lim and J. Chul Ye, "Geometric GAN," 2017, *arXiv:1705.02894*. [Online]. Available: http://arxiv.org/abs/1705.02894

[22] D. Tran, R. Ranganath, and D. M. Blei, "Hierarchical implicit models and likelihood-free variational inference," Jul. 2017, *arXiv:1702.08896*. [Online]. Available: http://arxiv.org/abs/1702.08896

[23] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," 2018, *arXiv:1805.08318*. [Online]. Available: http://arxiv.org/abs/1805.08318

[24] H. Jiang, Z. Chen, M. Chen, F. Liu, D. Wang, and T. Zhao, "On computation and generalization of generative adversarial networks under spectrum control," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–26.

[25] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," 2018, *arXiv:1809.11096*. [Online]. Available: http://arxiv.org/abs/1809.11096

[26] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," 2017, *arXiv:1701.00160*. [Online]. Available: http://arxiv.org/abs/1701.00160

[27] L. Mescheder, S. Nowozin, and A. Geiger, "The numerics of GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1825–1835.

[28] V. Nagarajan and J. Z. Kolter, "Gradient descent GAN optimization is locally stable," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5585–5595.

[29] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," 2016, *arXiv:1611.02163*. [Online]. Available: http://arxiv.org/abs/1611.02163

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

[31] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Citeseer, Univ. Toronto, Toronto, ON, Canada, 2009.

[32] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.

[33] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 4401–4410.

[34] W. Zhang, J. Sun, and X. Tang, "Cat head detection-how to effectively exploit shape and texture features," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2008, pp. 802–816.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[36] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*. [Online]. Available: http://arxiv.org/abs/1511.06434

[37] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2172–2180.

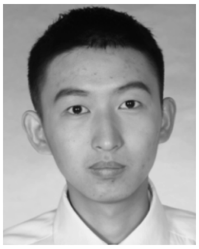
[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[39] G. L. Grinblat, L. C. Uzal, and P. M. Granitto, "Class-splitting generative adversarial networks," 2017, *arXiv:1709.07359*. [Online]. Available: http://arxiv.org/abs/1709.07359

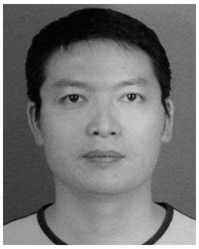
[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[41] D. C. Dowson and B. V. Landau, "The Fréchet distance between multivariate normal distributions," *J. Multivariate Anal.*, vol. 12, no. 3, pp. 450–455, Sep. 1982.

[42] D. Warde-Farley and Y. Bengio, "Improving generative adversarial networks with denoising feature matching," in *Proc. ICLR*, 2017.



XIAOPENG CHAO is currently pursuing the master's degree with the School of Information Engineering, Guangdong University of Technology, Guangzhou, China. His research interests include deep learning, computer vision, and generative models.



JIANGZHONG CAO received the Ph.D. degree from Sun Yat-sen University, in 2014. He is currently an Associate Professor with the Guangdong University of Technology, Guangzhou, China. He is also the Deputy Director of the Guangdong Provincial Key Laboratory of Intellectual Property Big Data. His research interests include image processing and machine learning.

YUQIN LU is currently pursuing the master's degree with the School of Information Engineering, Guangdong University of Technology, Guangzhou, China. His research interests include deep learning, computer vision, and multi-view image generation.

QINGYUN DAI received the Ph.D. degree in communication and electronic system from the South China University of Technology, Guangzhou, China, in 2001. She is currently a Professor with Guangdong Polytechnic Normal University, Guangzhou. She is also the Leader of the Guangdong Provincial Key Laboratory of Intellectual Property Big Data. Her research interests include the Internet of Things in manufacturing and image processing.

SHANGSONG LIANG received the Ph.D. degree from the University of Amsterdam, Amsterdam, The Netherlands, in 2014. He worked as a Visiting Postdoctoral Research Scientist with the University of Massachusetts Amherst and University College London. He is currently an Associate Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His expertise lies in the fields of information retrieval and text mining. He has extensively published his work in top-tier conferences and journals, including SIGIR, KDD, WWW, CIKM, AAAI, WSDM, NeurIPS, IEEE Transactions on Knowledge and Data Engineering, and *ACM Transactions on Information Systems*. He was a recipient of an Outstanding Reviewer Award in SIGIR 2017. He is also serving as an Editor for *Information Processing and Management*.