Omni-GAN: On the Secrets of cGANs and Beyond

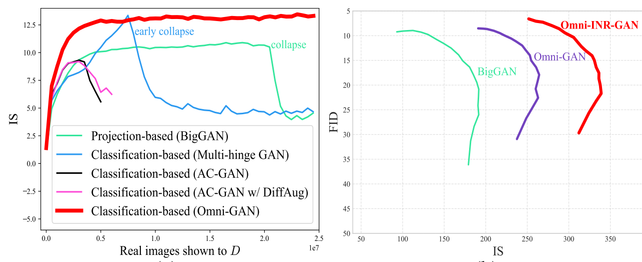
Peng Zhou[1], Lingxi Xie[2], Bingbing Ni[*,1], Cong Geng[1], Qi Tian[2]
[1]Shanghai Jiao Tong University, Shanghai 200240, China, [2]Huawei Inc.
{zhoupengcv, nibingbing, gengcong}@sjtu.edu.cn, 198808xc@gmail.com,
tian.qi1@huawei.com

Abstract

The conditional generative adversarial network (cGAN) is a powerful tool of generating high-quality images, but existing approaches mostly suffer unsatisfying performance or the risk of mode collapse. This paper presents ***Omni-GAN****, a variant of cGAN that reveals the devil in designing a proper discriminator for training the model. The key is to ensure that the discriminator receives strong supervision to perceive the concepts and moderate regularization to avoid collapse. Omni-GAN is easily implemented and freely integrated with off-the-shelf encoding methods (e.g., implicit neural representation, INR). Experiments validate the superior performance of Omni-GAN and Omni-INR-GAN in a wide range of image generation and restoration tasks. In particular, Omni-INR-GAN sets new records on the ImageNet dataset with impressive Inception scores of* ***262.85*** *and* ***343.22*** *for the image sizes of 128 and 256, respectively, surpassing the previous records by* ***100+*** *points. Moreover, leveraging the generator prior, Omni-INR-GAN can extrapolate low-resolution images to arbitrary resolution, even up to* $\times 60+$ *higher resolution. Code is available*[1].

1. Introduction

Generative Adversarial Network (GAN) [18] is a powerful tool for image generation [2, 28, 42] and domain adaptation [8, 24, 35, 64]. The big family of GAN can be roughly divided into two parts, *i.e.*, the unconditional GANs [30, 31] and conditional GANs (cGANs) [26, 41, 6, 67, 72, 69, 36], differing from each other in whether condition information is used for image generation. In practice, class-conditional GANs often suffer severe collapse when the number of categories is large. As shown in Fig. 1, all of BigGAN [6], Multi-hinge GAN [32], and AC-GAN [44] achieve high Inception scores, but the curves drop dramatically at some point of training. This makes the training unstable and thus early termination trick is used by the community [6].



(a) (b)

Figure 1: (a) The trend of Inception scores along the training procedure on CIFAR100, showing that Omni-GAN enjoys both high performance and a lower risk of mode collapse. (b) The tradeoff curves using the truncation trick to generate 128×128 images on ImageNet, where Omni-GAN and Omni-INR-GAN outperform BigGAN.

It has been noticed [29] that the instability of the training procedure is highly related to the discriminator, *i.e.*, the module that outputs a value indicating the reality of the generated image. The existing cGAN discriminators are roughly categorized into two types, namely, the projection-based [43, 6] and classification-based [44, 32] ones, according to whether the discriminator is required to output an explicit class label for each image. We find that, although the former choice (*i.e.*, a projection-based, with a weaker, implicit discriminator) is inferior to the latter in terms of the Inception score, the latter is prone to collapse (*e.g.*, in Fig. 1, Multi-hinge GAN achieves a high Inception score but collapses earlier).

This paper investigates the reason behind this phenomenon. We formulate the classification-based and projection-based discriminator into a multi-label classification framework, which offers us an opportunity to observe the advantages and disadvantages of them. As a result, we find that combining strong supervision (classification loss) and **moderate regularization** (to prevent it from quickly memorizing the training image set) is the best choice, where the GAN model enjoys high quality in image generation yet has a low risk of mode collapse. In practice, we use

[*]Corresponding author: Bingbing Ni.
[1]https://github.com/PeterouZh/Omni-GAN-PyTorch

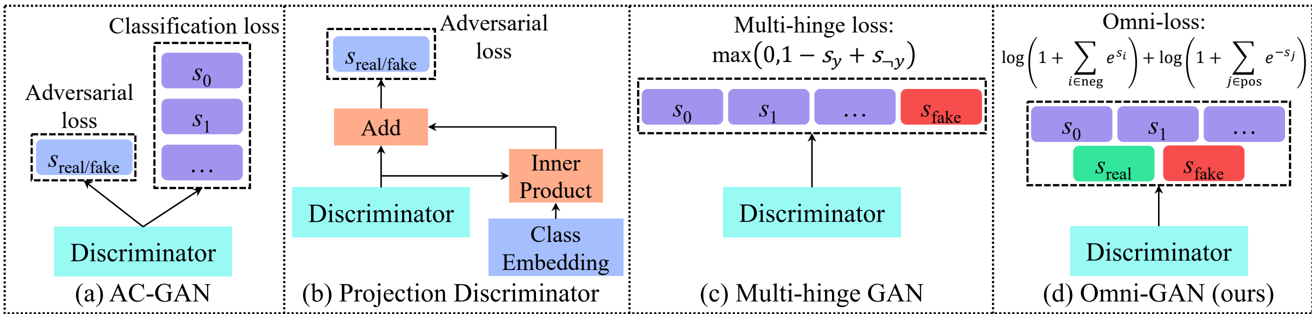


Figure 2: Different discriminator models for training a cGAN. Omni-loss supports to implement a classification-based cGAN or a projection-based cGAN, enabling us to fairly and intuitively explore the secrets behind them. Please refer to the texts in Sec. 2.1 and Sec. 3.1 for details.

weight decay as the choice of regularization, and our algorithm, named **Omni-GAN**, is easily implemented in any deep learning framework (adding a few lines of code beyond BigGAN). To show that our discovery generalizes to a wide range of cGAN models, we further integrate implicit neural representation (INR) [46, 40, 11], an off-the-shelf encoding method, into Omni-GAN. This not only improves the generation quality of Omni-GAN, but also enables it to generate images of any aspect ratio and any resolution, facilitating its application to downstream tasks.

We validate the advantages of our approach with extensive experiments on image generation and restoration. The image generation task is performed on CIFAR10, CIFAR100 [33], and ImageNet [14], three popular datasets. Omni-GAN surpasses the baselines in terms of the Fréchet Inception distance [21] and Inception score [54]. In particular, in generating 128×128 and 256×256 images on ImageNet, Omni-GAN achieves surprising Inception scores of 262.85 and 343.22, respectively, both of which surpassing the previous records by more than 100 points. The image restoration part involves colorization and single image super-resolution, where Omni-INR-GAN is more flexible than BigGAN and significantly outperforms other restoration methods like DIP [65] and LIIF [10], arguably because the prior learned by the generator is stronger.

We highlight the contributions of this paper as follows:

- The core discovery is that combining strong supervision and moderate regularization is the key to cGAN optimization. We achieve this goal easily with the proposed Omni-GAN framework.
- We integrate Omni-GAN into a recently published encoding named INR to validate its generalized ability and extend its range of applications.
- Omni-GAN achieves the state-of-the-art on the task of image generation on ImageNet, surpassing the prior best results by significant margins. It also requires fewer computational costs to gain the ability of high-quality image generation.

We will release the code to the public. We hope that our discovery can inspire the community in studying the principle of generative models and designing powerful algorithms.

2. Preliminaries

2.1. Class-conditional GANs

Conditional GAN (cGAN) [41] adds conditional information to the generator and discriminator of GANs [26, 25, 1, 2, 39, 52, 21, 27, 4, 55, 63, 12, 17, 37, 50, 9, 57]. There are some ways to incorporate class information into the generator, such as CBN [13], CIN [16, 22], CMConv [74], *etc.* There are also many ways to add class information to the discriminator. A simple way is to directly concatenate the class information with the input or features from some middle layers [15, 51, 68, 49, 53]. Next, we expound on several slightly complicated methods.

AC-GAN Auxiliary classifier GAN (AC-GAN) [44] uses an auxiliary classifier to enhance the standard GAN model (see Fig. 2a). In particular, the objective function consists of tow parts: the GAN loss, $\mathcal{L}_{\mathrm{GAN}}$, and the classification loss, $\mathcal{L}_{\mathrm{cls}}$:

$$\begin{aligned}\mathcal{L}_{\mathrm{GAN}} =& \mathbb{E}\left[\log P\left(\mathrm{g}=\text{real} \mid \boldsymbol{x}_{\text{real}}\right)\right]+ \\ & \mathbb{E}\left[\log P\left(\mathrm{g}=\text{fake} \mid \boldsymbol{x}_{\text{fake}}\right)\right],\end{aligned} \tag{1}$$

$$\mathcal{L}_{\mathrm{cls}}=\mathbb{E}\left[\log P\left(\mathrm{g}=c \mid \boldsymbol{x}_{\text{real}}\right)\right]+\mathbb{E}\left[\log P\left(\mathrm{g}=c \mid \boldsymbol{x}_{\text{fake}}\right)\right], \tag{2}$$

where g denotes the label of $\boldsymbol{x}$. $\boldsymbol{x}_{\text{real}}$ and $\boldsymbol{x}_{\text{fake}}$ represent a real image and a generated image respectively. The discriminator D of AC-GAN is trained to maximize $\mathcal{L}_{\mathrm{GAN}} + \mathcal{L}_{\mathrm{cls}}$, and the generator is trained to maximize $\mathcal{L}_{\mathrm{cls}} - \mathcal{L}_{\mathrm{GAN}}$. We will show that the discriminator loss of AC-GAN is not optimal (see Sec. 3.3).

Projection Discriminator Projection discriminator [43] incorporates class information into the discriminator of GANs in a projection-based way (see Fig. 2b). The mathematical form of the projection discriminator is given by

$$D(\boldsymbol{x},\boldsymbol{y})=\boldsymbol{y}^{\mathrm{T}}\boldsymbol{V} f_1\left(\boldsymbol{x};\theta_1\right)+f_2\left(f_1\left(\boldsymbol{x};\theta_1\right);\theta_2\right), \tag{3}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the input image and one-hot label vector respectively. $\boldsymbol{V}$ is a class embedding matrix, $f_1(\cdot;\theta_1)$ is a vector function, and $f_2(\cdot;\theta_2)$ is a scalar function. $\boldsymbol{V}, \theta_1, \theta_2$ are learned parameters of D. The discriminator D only outputs a scalar for each pair of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Multi-hinge GAN Multi-hinge GAN [32] belongs to classification-based cGANs. It uses a $C+1$ dimensional classifier as the discriminator, which is trained by a multi-class hinge loss (see Fig. 2c). Let the classifier be $S: \mathbb{X} \rightarrow \mathbb{R}^{C+1}$, the input image be $\boldsymbol{x}$, and the class label be $y \in \{0, 1, \ldots, C-1\}$. We use $\boldsymbol{s} = S(\boldsymbol{x})$ to denote the score vector of input image $\boldsymbol{x}$. The C-th element of $\boldsymbol{s}$, $s_C(\cdot)$, indicates the score corresponding to the fake (with indexing starting at 0). The discriminator loss is given by

$$\begin{aligned} \mathcal{L}_D = &\mathbb{E}_{(\boldsymbol{x},y)\sim p_{\mathrm{d}}}\left[\max\left(0, 1 - s_y(\boldsymbol{x}) + s_{\neg y}(\boldsymbol{x})\right)\right] \\ &+ \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}, y\sim p_{\mathrm{d}}}\left[\max\left(0, 1 - s_C(G(\boldsymbol{z}, y)) + s_{\neg C}\left(G(\boldsymbol{z}, y)\right)\right)\right], \end{aligned} \tag{4}$$

where $s_y(\boldsymbol{x})$ denotes the element y of vector $\boldsymbol{s}$, and $s_{\neg y}(\boldsymbol{x}) = \max_{k \neq y} s_k(\boldsymbol{x}), k \in \{0, 1, \ldots, C\} \backslash \{y\}$, represents the highest score except $s_y(\boldsymbol{x})$.

2.2. Implicit Neural Representation

Images are usually represented by a set of pixels with fixed resolution. A popular method named implicit neural representation (INR) is prevalent in the 3D field [46, 40, 11, 3, 19, 7, 48, 23]. Recently, people introduced the INR method to 2D images [10, 59, 58, 5, 60]. The INR of an image directly maps (x, y) coordinates to image's RGB pixel values. Since the coordinates are continuous, once we get the INR of an image, we can get images of arbitrary resolutions by sampling different numbers of coordinates.

2.3. Unified Loss for Feature Learning

There is a unified perspective for classification tasks. We denote the positive score set as $\mathbb{S}_{\mathrm{pos}} = \{s_1^{(p)}, \cdots, s_K^{(p)}\}$, and negative score set as $\mathbb{S}_{\mathrm{neg}} = \{s_1^{(n)}, \cdots, s_L^{(n)}\}$, respectively. Sun *et al.* [62] proposed a unified loss to maximize $s^{(p)}$ as well as to minimize $s^{(n)}$. The loss is defined as

$$\begin{aligned} \mathcal{L}_{\mathrm{uni}} &= \log\left[1 + \sum_{s_i^{(n)} \in \mathbb{S}_{\mathrm{neg}}} \sum_{s_j^{(p)} \in \mathbb{S}_{\mathrm{pos}}} e^{\left(\gamma\left(s_i^{(n)} - s_j^{(p)} + m\right)\right)}\right] \\ &= \log\left[1 + \sum_{s_i^{(n)} \in \mathbb{S}_{\mathrm{neg}}} e^{\left(\gamma\left(s_i^{(n)} + m\right)\right)} \sum_{s_j^{(p)} \in \mathbb{S}_{\mathrm{pos}}} e^{\left(\gamma\left(-s_j^{(p)}\right)\right)}\right], \end{aligned} \tag{5}$$

where γ stands for a scale factor, and m for a margin between positive and negative scores. Eq. (5) can be converted into triplet loss [56] or softmax with the cross-entropy loss [62].

3. Omni-GAN

3.1. Omni-GAN and One-sided Omni-GAN

We commence from defining the omni-loss. Based on this loss, we design two versions of cGANs: Omni-GAN, being a classification-based cGAN, and one-sided Omni-GAN, being a projection-based cGAN. These two cGANs enable us to fairly and intuitively explore the secrets behind classification-based cGANs and projection-based cGANs.

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote an image and its multi-label vector respectively. S is a classifier. Suppose that there are K positive labels and L negative labels. Then $\boldsymbol{s} = S(\boldsymbol{x})$ is a $K+L$ dimensional score vector. The omni-loss is defined as

$$\mathcal{L}_{\mathrm{omni}}(\boldsymbol{x}, \boldsymbol{y}) = \log\left(1 + \sum_{i \in \mathbb{I}_{\mathrm{neg}}} e^{s_i(\boldsymbol{x})}\right) + \log\left(1 + \sum_{j \in \mathbb{I}_{\mathrm{pos}}} e^{-s_j(\boldsymbol{x})}\right), \tag{6}$$

where $\mathbb{I}_{\mathrm{neg}}$ is a set consisting of indexes of negative scores (*i.e.*, $|\mathbb{I}_{\mathrm{neg}}| = L$), and $\mathbb{I}_{\mathrm{pos}}$ consists of indexes of positive scores (*i.e.*, $|\mathbb{I}_{\mathrm{pos}}| = K$). $s_k(\boldsymbol{x})$ represents the element k of vector $\boldsymbol{s}$. [61] shows that Eq. (6) is a special case of Eq. (5). We provide a detailed derivation from Eq. (5) to Eq. (6) in Appendix A. Next, we introduce two versions of Omni-GAN by setting different labels for the omni-loss.

- **The classification-based Omni-GAN.**

We first elucidate the loss of the discriminator for Omni-GAN. The discriminator loss consists of two parts, one for $\boldsymbol{x}_{\mathrm{real}}$ (drawn from the training data), and the other for $\boldsymbol{x}_{\mathrm{fake}}$ (drawn from the generator). For $\boldsymbol{x}_{\mathrm{real}}$, its multi-label vector is given by

$$\boldsymbol{y}_{\mathrm{real}} = [\underbrace{-1, \ldots, 1_{\mathrm{gt}}, \ldots, -1}_{C}, \underbrace{1_{\mathrm{real}}, -1}_{2}], \tag{7}$$

whose dimension is $C+2$, with C being the number of classes of the training dataset. 1_{gt} is 1 if its index in the vector is equal to the ground truth label of $\boldsymbol{x}_{\mathrm{real}}$, otherwise -1. We use 1 to denote the corresponding score belongs to the positive set, and -1 to the negative set. The multi-label vector of $\boldsymbol{x}_{\mathrm{fake}}$ is also a $C+2$ dimensional vector:

$$\boldsymbol{y}_{\mathrm{fake}} = [\underbrace{-1, \ldots, -1, \ldots, -1}_{C}, \underbrace{-1, 1_{\mathrm{fake}}}_{2}], \tag{8}$$

where only the last element is 1.

According to Eq. (6), (7), and (8), we define the discriminator loss as

$$\begin{aligned} \mathcal{L}_D = &\mathbb{E}_{\boldsymbol{x}_{\mathrm{real}}\sim p_{\mathrm{d}}}\left[\mathcal{L}_{\mathrm{omni}}\left(\boldsymbol{x}_{\mathrm{real}}, \boldsymbol{y}_{\mathrm{real}}\right)\right] \\ &+ \mathbb{E}_{\boldsymbol{x}_{\mathrm{fake}}\sim p_{\mathrm{g}}}\left[\mathcal{L}_{\mathrm{omni}}\left(\boldsymbol{x}_{\mathrm{fake}}, \boldsymbol{y}_{\mathrm{fake}}\right)\right], \end{aligned} \tag{9}$$

where p_{d} is the training data distribution, and p_{g} is the generated data distribution. It is obvious that the discriminator

D actually acts as a multi-label classifier, which takes as input $\boldsymbol{x}$, and outputs a score vector $\boldsymbol{s} = D(\boldsymbol{x})$.

The generator attempts to fool the discriminator into believing its samples are real. To this end, its multi-label is set to be

$$\boldsymbol{y}_{\text{fake}}^{(\text{G})} = [\underbrace{-1, \ldots, 1_{\text{G}}, \ldots, -1}_{C}, \underbrace{1_{\text{real}}, -1}_{2}], \tag{10}$$

which is the same as $\boldsymbol{y}_{\text{real}}$ defined in Eq. (7). 1_{G} is 1 if its index in the vector is equal to the label adopted by the generator to generate $\boldsymbol{x}_{\text{fake}}$, otherwise -1. The generator loss is then given by

$$\mathcal{L}_G = \mathbb{E}_{\boldsymbol{x}_{\text{fake}} \sim p_{\text{g}}} \left[\mathcal{L}_{\text{omni}} \left(\boldsymbol{x}_{\text{fake}}, \boldsymbol{y}_{\text{fake}}^{(\text{G})} \right) \right]. \tag{11}$$

- **The projection-based (one-sided) Omni-GAN.**

We imitate the way how the projection-based discriminator [43] utilizes class labels (see Eq. (3)), and design a projection-based variant of Omni-GAN, named one-sided Omni-GAN, which does not fully utilize the class supervision.

It is easy to implement one-sided Omni-GAN: only slightly modify the multi-label vector, $\boldsymbol{y}$. Following the setting above, the multi-label vector for $\boldsymbol{x}_{\text{real}}$ is set to be

$$\boldsymbol{y}_{\text{real}} = [\underbrace{0, \ldots, 1_{\text{gt}}, \ldots, 0}_{C}, \underbrace{1, 0}_{2}], \tag{12}$$

where 1_{gt} is 1 if its index in the vector is equal to the ground truth label of $\boldsymbol{x}_{\text{real}}$, otherwise 0. And 0 means that the corresponding score will be ignored when calculating the omni-loss. The multi-label vector for $\boldsymbol{x}_{\text{fake}}$ is given by

$$\boldsymbol{y}_{\text{fake}} = [\underbrace{0, \ldots, -1_{\text{G}}, \ldots, 0}_{C}, \underbrace{-1, 0}_{2}], \tag{13}$$

where -1_{G} is -1 if its index in the vector is equal to the label adopted by the generator to generate $\boldsymbol{x}_{\text{fake}}$, otherwise 0. The discriminator loss is the same as that defined in Eq. (9).

For generator, its multi-label vector for $\boldsymbol{x}_{\text{fake}}$ is

$$\boldsymbol{y}_{\text{fake}}^{(\text{G})} = [\underbrace{0, \ldots, 1_{\text{G}}, \ldots, 0}_{C}, \underbrace{1, 0}_{2}], \tag{14}$$

where 1_{G} is 1 if its index in the vector is equal to the label adopted by the generator to generate $\boldsymbol{x}_{\text{fake}}$, otherwise 0. The generator loss is the same as that defined in Eq. (11).

In summary, we introduce two versions of Omni-GAN by setting different multi-label vector for the omni-loss (defined in Eq. (6)). It is easy to implement these two GANs in practice: as shown in Fig. 2d, first, let the discriminator output a vector instead of a scalar; second, apply the omni-loss to the output vector.

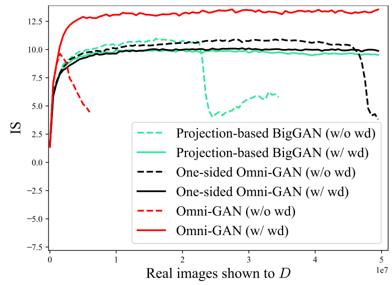


Figure 3: IS on CIFAR100. "wd" stands for weight decay. The combination of strong supervision and weight decay is crucial. Weight decay effectively alleviates the collapse problem of strongly supervised cGANs (Omni-GAN) so that the cGANs enjoy superior performance from strong supervision. On the other hand, weight decay may slightly impair the performance of weakly supervised cGANs (*e.g.*, projection-based BigGAN, one-sided Omni-GAN).

3.2. The Devil Lies in Combination of Strong Supervision and Moderate Regularization

We conducted control experiments on CIFAR100 [33], and compared Omni-GAN and one-sided Omni-GAN with a projection-based cGAN, namely BigGAN [6]. As shown in Fig. 3, One-sided Omni-GAN is on par with BigGAN in terms of IS, indicating that one-sided Omni-GAN is indeed a projection-based cGAN. We can see from Eq. (12), (13) and (14) that one-sided Omni-GAN does not make full use of the class label's supervision. Those elements with label 0 in the output vector of the discriminator are ignored (*i.e.*, they are not used to calculate the omni-loss, meaning these elements will not get gradients when backpropagation). Therefore, we think that the projection-based cGAN is an implicit and weaker cGAN in the sense that it does not make full use of the supervision of the class label.

On the other hand, as a classification-based cGAN, Omni-GAN makes full use of class supervision (strong supervision). However, it suffers a severe collapse in the initial stages of training. As shown in Fig. 3, the IS of Omni-GAN shows a significant upward compared to the projection-based cGAN but drops dramatically when about 1M real images (20 epoch) are shown to the discriminator.

Our core discovery is that a moderate regularization effectively prevents the early collapse of classification-based cGANs. In practice, we use weight decay [34] as the choice of regularization. We call weight decay moderate regularization in that it does not introduce considerable computational overhead as other regularizations do, such as gradient penalty [20, 38, 66]. Therefore, Omni-GAN can be trained efficiently on large-scale datasets such as ImageNet. As shown in Fig. 3, combined with weight decay, Omni-GAN has greatly improved its IS compared to BigGAN. Moreover, we observe the projection-based cGAN, BigGAN, also collapses after long training. Weight decay is also effective for alleviating the collapse of BigGAN.

Note that we are not the first to use weight decay in GANs. [73] applies weight decay to unconditional GANs. However, our main contribution is to emphasize that the combination of strong supervision and weight decay is the key to cGANs. In fact, combining weight decay with weakly supervised cGANs even hurts performance. As shown in Fig. 3, both projection-based BigGAN and one-sided Omni-GAN suffer performance degradation after combined with weight decay.

In summary, we claim that the combination of strong supervision and weight decay is critical for cGANs. Strong supervision helps boost the performance of cGANs but causes severe early collapse. Weight decay effectively alleviates early collapse, so that cGANs can fully enjoy the benefits of strong supervision.

3.3. Comparison to Previous Approaches

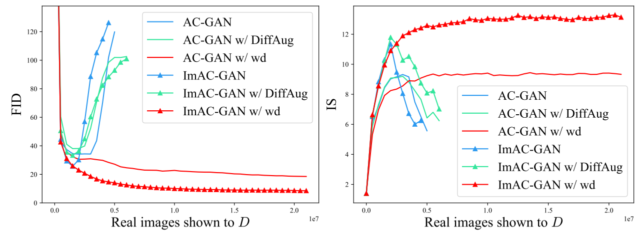
We study another well-known classification-based cGAN, AC-GAN [44], and show its results in Fig. 4. AC-GAN also suffers severe early collapse like Omni-GAN does. One possible explanation for the early collapse of classification-based cGANs is that the discriminator overfits the training data [29]. Therefore, we study whether data augmentation is effective to alleviate the early collapse. As shown in Fig. 4, AC-GAN combined with differentiable data augmentation (DiffAug) [70][2] still cannot avoid early collapse. However, weight decay is still very effective in alleviating the early collapse of AC-GAN.

We emphasize that weight decay prevents overfitting of the discriminator at the model level, and data augmentation does at the data level. We will show in experiments that both methods are effective for improving the performance of cGANs. However, we empirically find that weight decay is almost 100% effective in preventing the early collapse of classification-based cGANs, but data augmentation is not always effective.

Next, we investigate whether spectral normalization (SN) [42] and gradient penalty [20, 38, 66] will alleviate early collapse. Because the network architectures of the generator and discriminator we used in our experiments employ SN by default, Omni-GAN and AC-GAN still suffer severe early collapse, indicating that SN cannot alleviate the collapse problem of classification-based cGANs. In addition, we have empirically found that gradient penalty cannot help classification-based cGANs avoid early collapse (refer to Appendix B for experimental results).

Finally, by comparing AC-GAN and Omni-GAN's loss functions, we found that the original AC-GAN still has room for improvement. Due to space limitations, we put the details in Appendix C. We name the improved AC-GAN ImAC-GAN. As shown in Fig. 4, the performance of ImAC-GAN is significantly better than that of AC-GAN, and is comparable to that of Omni-GAN (refer to Sec. 4.1).

[2]We did not choose ADA augmentation because ADA needs to design different overfitting heuristics for different losses.



(a) FID on CIFAR100 (b) IS on CIFAR100

Figure 4: AC-GAN suffers a severe collapse in the initial stage of training. "wd" stands for weight decay, DiffAug for differentiable data augmentation. Weight decay effectively alleviates early collapse, but data augmentation does not. ImAC-GAN means an improved version of AC-GAN. Please refer to Sec. 3.3 for details.

(a) Grayscale image (b) Restored image (DGP w/ BigGAN) (c) Restored image (DGP w/ Omni-INR-GAN) (d) Ground truth

Figure 5: Colorization. (a) degraded input. (b) and (c) GAN inversion-based colorization algorithms. BigGAN only colorizes a square image patch. Omni-INR-GAN directly colorizes the entire image.

To sum up, our results reveals that fully utilizing the supervision can improve performance of cGANs, but at the risk of early collapse. This work offers a practical way (weight decay) to overcome the collapse issue, so that the trained model enjoys both superior performance and safe optimization.

3.4. Omni-INR-GAN: Being Friendly to Downstream Tasks

Omni-GAN is easily implemented and freely integrated with off-the-shelf encoding methods. We derive Omni-INR-GAN by integrating implicit neural representation (INR) [46, 40, 11] into Omni-GAN. Omni-INR-GAN employs INR to enhance the generator's output layer. Due to limited space, we detail technical details in Appendix D.

Omni-INR-GAN has the ability to output images with any aspect ratio and resolution. Thus it is friendly to downstream tasks like image restoration. Fig. 5 shows an example of combining Omni-INR-GAN with DGP [45] for colorization. As shown in Fig. 5 (b), the DGP with BigGAN model only colorize a square patch in the original image. This is because BigGAN only generates fixed-size images, which is inflexible for downstream tasks. On the other hand, Omni-INR-GAN is able to generate images with any aspect ratio and any resolution easily so as to directly handle the entire degraded image. Because the generator has seen considerable numbers of natural images, it owns a wealth of

No.	Method	CIFAR10			CIFAR100		
		FID ↓	IS ↑	Collapse?	FID ↓	IS ↑	Collapse?
-	FQ-GAN [71]	6.16	9.16	No	8.23	10.62	No
-	Multi-hinge [32]	6.22	9.55	No	14.62	13.35	Yes
-	ADA [29]	**2.67**†	10.06†	-	-	-	-
0	BigGAN, ~~wd~~ [6]	7.27	9.19	Yes	10.12	10.96	Yes
1	BigGAN, wd	7.77	9.29	No	13.73	9.98	No
2	BigGAN, DiffAug, ~~wd~~	5.46	9.28	No	8.63	10.69	No
3	BigGAN, DiffAug, wd	7.97	9.61	No	13.97	9.79	No
4	AC-GAN, ~~wd~~ [44]	7.11	9.43	Yes	34.19	9.31	Yes
5	AC-GAN, wd	8.03	9.58	No	16.41	9.74	No
6	AC-GAN, DiffAug, ~~wd~~	5.71	9.76	No	38.01	9.19	Yes
7	AC-GAN, DiffAug, wd	8.98	9.75	No	14.91	11.01	No
8	ImAC-GAN, ~~wd~~	6.62	9.46	Yes	25.64	11.34	Yes
9	ImAC-GAN, wd	5.63	9.72	No	8.11	13.38	No
10	ImAC-GAN, DiffAug, ~~wd~~	4.57	9.85	No	33.30	11.78	Yes
11	ImAC-GAN, DiffAug, wd	6.66	10.02	No	8.45	14.54	No
12	Omni-GAN, ~~wd~~	7.75	9.74	Yes	26.51	11.45	Yes
13	Omni-GAN, wd	5.57	9.79	No	8.41	13.30	No
14	Omni-GAN, DiffAug, ~~wd~~	7.13	9.86	No	39.90	10.66	Yes
15	Omni-GAN, DiffAug, wd	7.83	**10.37**	No	11.39	**15.37**	No
16	Omni-INR-GAN, ~~wd~~	8.59	9.74	Yes	53.29	9.05	Yes
17	Omni-INR-GAN, wd	5.25	9.74	No	7.63	13.90	No
18	Omni-INR-GAN, DiffAug, ~~wd~~	75.75	5.80	Yes	53.88	11.06	Yes
19	Omni-INR-GAN, DiffAug, wd	4.32	10.03	No	**6.70**	14.15	No

Table 1: FID and IS on CIFAR10 and CIFAR100. † indicates quoted from the paper. "wd" stands for applying weight decay, and "~~wd~~" for not applying weight decay. DiffAug means differentiable data augmentation. FID and IS are computed using 50K training and 50K generated images with the TensorFlow-based pre-trained Inception-V3 model. Please refer to Sec. 4.1 for detailed analysis.

prior knowledge. As shown in Fig. 5 (c), utilizing the generator prior helps get a plausible color image.

4. Experiments

4.1. Evaluation on CIFAR

We compare a projection-based cGAN, BigGAN, and several classification-based cGANs, including AC-GAN, ImAC-GAN, Omni-GAN, and Omni-INR-GAN. We also study the effects of data augmentation and weight decay on these methods. Results are summarized in Table 1.

First, let us focus on the projection-based cGAN, BigGAN. We found that data augmentation helps improve the performance of BigGAN, but weight decay cannot, even being harmful. This is reasonable because BigGAN belongs to projection-based cGANs, whose discriminator is essentially a weak implicit classifier. Weight decay constrains the fitting ability of the discriminator (*i.e.*, a weak network). Moreover, the supervision for the discriminator is too weak (*i.e.*, a weak supervision). As such, the performance is naturally not good.

Next, we compare AC-GAN and ImAC-GAN, which belong to classification-based cGANs. ImAC-GAN has a clear and consistent improvement over AC-GAN in all experiments. This improvement is achieved by slightly modifying the loss function of AC-GAN to be consistent with Omni-GAN (refer to Appendix C for technical details).

Third, ImAC-GAN and Omni-GAN are both superior classification-based cGANs, and their performance is comparable. ImAC-GAN uses cross-entropy as the classification loss, so it only supports single-label classification. However, the omni-loss used by Omni-GAN naturally supports multi-label classification. Therefore when the image owns multiple positive labels, Omni-GAN is more flexible than ImAC-GAN. We give an example of using Omni-GAN to generate images with multiple positive labels in Appendix E.

We find that training collapse is more likely to appear on CIFAR100 that owns more classes. For example, all classification-based cGANs, including AC-GAN, ImAC-GAN, Omni-GAN, and Omni-INR-GAN, collapsed on CIFAR100 (shown in Table 1, No. #4, #8, #12, #16). Even if equipped with data augmentation, they still collapsed (No. #6, #10, #14, #18). However, weight decay effectively alleviates the training collapse of these methods. On CIFAR10, data augmentation can alleviate the collapse issue of classification-based cGANs (No. #6, #10, #14). We think a possible reason is that CIFAR10 has a relatively small number of categories, and each category has $5,000$ training images. Combining with data augmentation can effectively prevent the discriminator from overfitting the training data. An anomaly is that data augmentation cannot prevent the collapse of Omni-INR-GAN on CIFAR10 (#18). We found in our experiments that data augmentation seems to be somewhat exclusive with Omni-INR-GAN. We guess that one possible reason is that some random shift operations in data augmentation destroy some prior information to be learned by Omni-INR-GAN from the coordinates.

Last but not least, we have tried to combine Omni-GAN with the network architecture of StyleGAN2 [31], but failed. StyleGAN2 employs some technologies like equalized learning rate [28] (explicitly scales the weights at runtime) during the training process. We found that these techniques seem to conflict with weight decay, which is the key for classification-based cGANs to avoid early collapse. Combining strong classification loss with StyleGANs while avoiding early collapse needs further exploration.

4.2. Evaluation on ImageNet

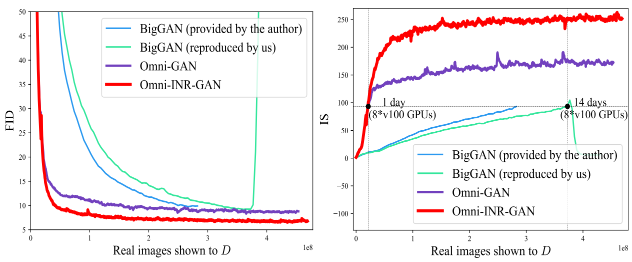
ImageNet [14] is a large dataset with 1000 number of classes and approximate 1.2M training data. We trained BigGAN[3], Omni-GAN, and Omni-INR-GAN on ImageNet 128×128 and 256×256, respectively. The results are shown in Table 2, and the convergence curves are shown in Fig. 6 (please refer to Appendix G for more results).

As shown in Table 2 and Fig. 6, Omni-GAN shows significant advantages over BigGAN, both in terms of convergence speed and final performance. For example, Omni-GAN only took one day to reach the IS of BigGAN trained for 14 days on ImageNet 128×128. Besides, its IS is almost twice that of BigGAN, namely 190.94 vs. 104.57.

[3]https://github.com/ajbrock/BigGAN-PyTorch.

Method	ImageNet 128 × 128				ImageNet 256 × 256			
	FID (train) ↓	FID (val) ↓	IS ↑	G Params	FID (train) ↓	FID (val) ↓	IS ↑	G Params
CR-BigGAN† [69]	6.66	-	-	-	-	-	-	-
S3GAN† [36]	7.70	-	83.10	-	-	-	-	-
BigGAN‡ [6]	9.77	9.96	93.09	70.43M	-	-	-	-
BigGAN⋆	9.19	9.18	104.57	70.43M	9.95	9.88	187.60	82.10M
Omni-GAN	8.30(0.89 ↓)	8.93(0.25 ↓)	190.94(86.37 ↑)	70.43M	6.03(3.92 ↓)	6.83(3.05 ↓)	304.05(116.45 ↑)	82.10M
Omni-INR-GAN	**6.53**(**2.66** ↓)	**7.99**(**1.19** ↓)	**262.85**(**158.28** ↑)	70.52M	**4.93**(**5.02** ↓)	**6.36**(**3.52** ↓)	**343.22**(**155.62** ↑)	82.19M

Table 2: FID and IS on ImageNet dataset. Omni-GAN achieves consistent improvements in terms of FID and IS compared to BigGAN. Omni-INR-GAN improves the IS to 2.5 times compared with BigGAN on ImageNet 128 × 128, with almost the same number of parameters. † stands for quoting from the paper, ‡ for using the model provided by the author, and ⋆ for reproducing BigGAN by us. FID and IS are computed using 50K generated images. The training and validation data are utilized as the reference distribution for the computing of FID, respectively.



(a) FID on ImageNet 128 × 128 (b) IS on ImageNet 128 × 128

Figure 6: Convergence curves on ImgeNet. Both Omni-GAN and Omni-INR-GAN converge faster than the projection-based cGAN, BigGAN. In particular, Omni-GAN only took one day to reach the IS of BigGAN trained for 14 days. Omni-INR-GAN consistently outperforms BigGAN and Omni-GAN. Its IS is 2.5 times higher than that of BigGAN (namely 262.85 *vs.* 104.57).

Omni-INR-GAN consistently outperforms BigGAN and Omni-GAN. As shown in Table 2, on ImageNet 128 × 128, the IS of Omni-INR-GAN is 2.5 times that of BigGAN. We also show the number of parameters of the generator in Table 2. The number of parameters of Omni-INR-GAN is on par with that of BigGAN and Omni-GAN, indicating that the improvement does not lie in the number of parameters. We think the possible reason is that Omni-INR-GAN introduces coordinates as input, which helps the generator learn some prior knowledge of the natural images (for example, the sky often appears in the upper part of an image, and the grass often appears in the lower part on the contrary).

In summary, the significant improvement of Omni-GAN lies in the combination of strong supervision and weight decay. Strong supervision helps boost the performance of cGANs, but it causes the training to collapse earlier. Weight decay effectively alleviates the collapse problem, so that cGANs fully enjoy the benefits from strong supervision.

4.3. Application to Image-to-Image Translation

We improve the mIoU of SPADE [47] from 62.21 to 65.07 by only using Omni-GAN loss and weight decay for the discriminator (please refer to Appendix H for details). We believe that the improvement comes from the improved ability of the discriminator in distinguishing different classes, so that the generator receives better guidance and thus produces images with richer semantic information.

	BigGAN	Omni-GAN	Omni-INR-GAN
PSNR↑	25.68	26.35	**29.36**
SSIM↑	85.17%	89.36%	**92.74%**

Table 3: Image reconstruction results using pre-trained GAN models. High performance shows the potential to be applied to downstream tasks.

4.4. Application to Downstream Tasks

Colorization. Fig. 7 shows an example of using the pre-trained BigGAN and Omni-INR-GAN to colorize images, respectively. See Appendix I for technical details. Omni-INR-GAN directly colorizes the entire image because it has the ability to output images of any resolution. However, BigGAN cannot do this. Another interesting phenomenon is that Omni-INR-GAN gradually overlays colors on the corresponding objects, indicating that the finetuning process is mining GAN's prior information.

Prior Enhanced Super-Resolution Fig. 8 shows an example of using pre-trained Omni-INR-GAN for super-resolution. We deliver two messages. First, Omni-INR-GAN can extrapolate low-resolution images to any resolution. Fig. 8 (f) shows the results with upsampling scales of ×1, ×4.6, and even ×63.5. Second, Omni-INR-GAN has a wealth of prior knowledge, which helps complement the missing semantics of the input. For example, for the extremely low-resolution zebra image shown in Fig. 8 (b), the image details are severely missing. In this case, although LIIF [10] has the ability to up-sample images at any scale, it cannot fill in the missing stripes of the zebra, as shown in Fig. 8 (c).

In Fig. 8 (d), DIP [65] also fails. DIP only leverages the input image and the structure of a ConvNet as the image prior. It cannot work when the input image's resolution is

Figure 7: Colorization example. Top: DGP with BigGAN. Bottom: DGP with Omni-INR-GAN. BigGAN only colorizes a square image patch. Omni-INR-GAN directly colorizes the entire image. Omni-INR-GAN gradually overlays colors on the corresponding objects, indicating that the finetuning process is mining GAN's prior information.

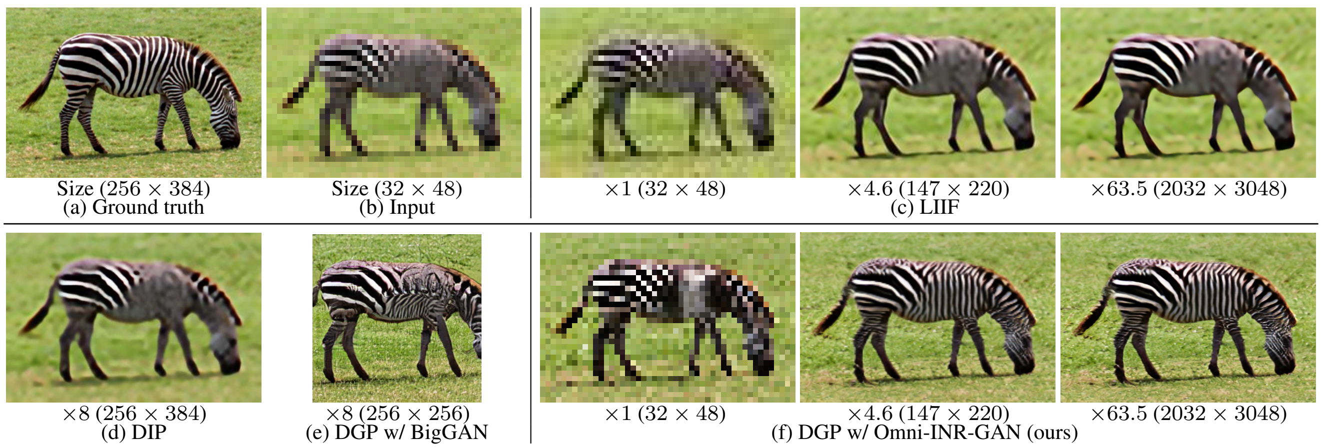


Figure 8: Super-resolution using Omni-INR-GAN's prior, at any scale ($\times$1-$\times$60+). (b) input image with low resolution. (c) LIIF [10] can extrapolate the input image to any scale, but it cannot add semantic details, so the result is still blurred. (d) DIP [65] also failed because the input image resolution is too low. (e) DGP [45] with BigGAN must crop the input and upsamples the cropped patch to a fixed size, which is inflexible. (f) Omni-INR-GAN has the ability to upsample the input image to any scale and also adds rich semantic details (clearer foreground and background compared to other methods). Please see the video demo in the supplementary material.

too low. DGP [45] with BigGAN can only handle fixed-size images, limiting its practical application (Fig. 8 (e)). However, DGP with Omni-INR-GAN can super-resolve the entire image and extrapolate the input at any scale. In Fig. 8 (f), by mining Omni-INR-GAN's prior, the missing zebra stripes are complemented, and a clear foreground and background are obtained. Even the shadow of the zebra is clearly visible.

In Table 3, we quantitatively compare different pre-trained GAN models for image reconstruction. Please refer to Appendix I for details. Omni-INR-GAN outperforms BigGAN and Omni-GAN by a large margin, showing its potential for downstream tasks.

5. Conclusion

This paper presents an elegant and practical solution to training effective conditional GAN models. The key discovery is that strong supervision can largely improve the upper-bound of image generation quality, but it also makes the model collapse earlier. We design the **Omni-GAN** algorithm that equips the classification-based loss with regularization (in particular, weight decay) to alleviate collapse. Our algorithm achieves notable performance gain in various scenarios including image generation and restoration. Our research implies that there may be more 'secrets' in optimizing cGAN models. We look forward to applying the proposed algorithm and pre-trained models to more scenarios and investigating further properties to improve cGAN.

Acknowledgements

This work was supported by National Science Foundation of China (U20B2072, 61976137). The first author would like to thank Lin Liu for instructive discussions.

References

[1] Martin Arjovsky and Léon Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv:1701.04862 [cs, stat]*, 2017. 2

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *ICML*, 2017. 1, 2

[3] Matan Atzmon and Yaron Lipman. SAL: Sign Agnostic Learning of Shapes from Raw Data. *arXiv:1911.10414 [cs]*, 2020. 3

[4] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. *arXiv:1811.10597 [cs]*, 2018. 2

[5] Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. X-Fields: Implicit Neural View-, Light- and Time-Image Interpolation. *arXiv:2010.00450 [cs]*, 2020. 3

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv:1809.11096*, 2018. 1, 4, 6, 7

[7] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction. *arXiv:2003.10983 [cs]*, 2020. 3

[8] Minghao Chen, Shuai Zhao, Haifeng Liu, and Deng Cai. Adversarial-Learned Loss for Domain Adaptation. In *AAAI*, 2020. 1

[9] Tianlong Chen, Yu Cheng, Zhe Gan, Jingjing Liu, and Zhangyang Wang. Ultra-Data-Efficient GAN Training: Drawing A Lottery Ticket First, Then Training It Toughly. *arXiv:2103.00397 [cs]*, 2021. 2

[10] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning Continuous Image Representation with Local Implicit Image Function. *arXiv:2012.09161 [cs]*, 2020. 2, 3, 7, 8

[11] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. *arXiv:1812.02822 [cs]*, 2019. 2, 3, 5

[12] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse Image Synthesis for Multiple Domains. In *CVPR*, 2020. 2

[13] Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. In *NeurIPS*, 2017. 2

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, Miami, FL, 2009. 2, 6

[15] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. *arXiv:1506.05751 [cs]*, 2015. 2

[16] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A Learned Representation For Artistic Style. In *ICLR*, 2017. 2

[17] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. AutoGAN: Neural Architecture Search for Generative Adversarial Networks. In *ICCV*, 2019. 2

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *NeurIPS*, 2014. 1

[19] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for Learning Shapes. *arXiv:2002.10099 [cs, stat]*, 2020. 3

[20] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017. 4, 5

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *NeurIPS*, 2017. 2

[22] Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In *ICCV*, 2017. 2

[23] Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local Implicit Grid Representations for 3D Scenes. *arXiv:2003.08981 [cs]*, 2020. 3

[24] Xiang Jiang, Qicheng Lao, Stan Matwin, and Mohammad Havaei. Implicit Class-Conditioned Domain Alignment for Unsupervised Domain Adaptation. In *ICML*, 2020. 1

[25] Alexia Jolicoeur-Martineau. The relativistic discriminator: A key element missing from standard GAN. In *ICLR*, 2019. 2

[26] Minguk Kang and Jaesik Park. ContraGAN: Contrastive learning for conditional image generation. In *NeurIPS*, 2020. 1, 2

[27] Animesh Karnewar and Oliver Wang. MSG-GAN: Multi-Scale Gradient GAN for Stable Image Synthesis. *arXiv:1903.06048 [cs, stat]*, 2019. 2

[28] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv:1710.10196 [cs, stat]*, 2017. 1, 6

[29] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training Generative Adversarial Networks with Limited Data. *arXiv:2006.06676 [cs, stat]*, 2020. 1, 5, 6

[30] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *CVPR*, 2019. 1

[31] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN. *arXiv:1912.04958 [cs, eess, stat]*, 2019. 1, 6

[32] Ilya Kavalerov, Wojciech Czaja, and Rama Chellappa. cGANs with Multi-Hinge Loss. *arXiv:1912.04216 [cs, stat]*, 2019. 1, 3, 6

[33] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 2, 4

[34] Anders Krogh and John A. Hertz. A Simple Weight Decay Can Improve Generalization. In *NeurIPS*, 1992. 4

[35] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional Adversarial Domain Adaptation. In *NeurIPS*, 2018. 1

[36] Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-Fidelity Image Generation With Fewer Labels. *arXiv:1903.02271 [cs, stat]*, 2019. 1, 7

[37] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least Squares Generative Adversarial Networks. *arXiv:1611.04076 [cs]*, 2016. 2

[38] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually Converge? In *ICML*, 2018. 4, 5

[39] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The Numerics of GANs. In *NeurIPS*, 2017. 2

[40] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. *arXiv:1812.03828 [cs]*, 2019. 2, 3, 5

[41] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv:1411.1784 [cs, stat]*, 2014. 1, 2

[42] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. *arXiv:1802.05957 [cs, stat]*, 2018. 1, 5

[43] Takeru Miyato and Masanori Koyama. cGANs with Projection Discriminator. In *ICLR*, 2018. 1, 2, 4

[44] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. *arXiv:1610.09585 [cs, stat]*, 2017. 1, 2, 5, 6

[45] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting Deep Generative Prior for Versatile Image Restoration and Manipulation. In *ECCV*, 2020. 5, 8

[46] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *arXiv:1901.05103 [cs]*, 2019. 2, 3, 5

[47] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic Image Synthesis with Spatially-Adaptive Normalization. In *CVPR*, 2019. 7

[48] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional Occupancy Networks. *arXiv:2003.04618 [cs]*, 2020. 3

[49] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible Conditional GANs for image editing. *arXiv:1611.06355 [cs]*, 2016. 2

[50] Guo-Jun Qi. Loss-Sensitive Generative Adversarial Networks on Lipschitz Densities. *arXiv:1701.06264 [cs]*, 2017. 2

[51] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative Adversarial Text to Image Synthesis. In *ICML*, 2016. 2

[52] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing Training of Generative Adversarial Networks through Regularization. In *NeurIPS*, 2017. 2

[53] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal Generative Adversarial Nets with Singular Value Clipping. In *ICCV*, 2017. 2

[54] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved Techniques for Training GANs. In *NeurIPS*, 2016. 2

[55] Edgar Schönfeld, Bernt Schiele, and Anna Khoreva. A U-Net Based Discriminator for Generative Adversarial Networks. In *CVPR*, 2020. 2

[56] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *CVPR*, 2015. 3

[57] Woohyeon Shim and Minsu Cho. CircleGAN: Generative Adversarial Learning across Spherical Circles. 2020. 2

[58] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *NeurIPS*, 2020. 3

[59] Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial Generation of Continuous Images. *arXiv:2011.12026 [cs]*, 2020. 3

[60] Kenneth O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2), 2007. 3

[61] Jianlin Su. Extending Cross-Entropy of Softmax to Multi-Label Classification. https://kexue.fm/archives/7359, 2020. 3

[62] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle Loss: A Unified Perspective of Pair Similarity Optimization. In *CVPR*, 2020. 3

[63] Yuesong Tian, Li Shen, Li Shen, Guinan Su, Zhifeng Li, and Wei Liu. AlphaGAN: Fully Differentiable Architecture Search for Generative Adversarial Networks. *arXiv:2006.09134 [cs, eess]*, 2020. 2

[64] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial Discriminative Domain Adaptation. *arXiv:1702.05464 [cs]*, 2017. 1

[65] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep Image Prior. *arXiv:1711.10925 [cs, stat]*, 2018. 2, 7, 8

[66] Jiqing Wu, Zhiwu Huang, Janine Thoma, Dinesh Acharya, and Luc Van Gool. Wasserstein divergence for gans. In *ECCV*, 2018. 4, 5

[67] Yan Wu, Jeff Donahue, David Balduzzi, Karen Simonyan, and Timothy Lillicrap. LOGAN: Latent Optimisation for Generative Adversarial Networks. *arXiv:1912.00953 [cs, stat]*, 2019. 1

[68] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. In *ICCV*, 2017. 2

[69] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency Regularization for Generative Adversarial Networks. In *ICLR*, 2020. 1, 7

[70] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable Augmentation for Data-Efficient GAN Training. *arXiv:2006.10738 [cs]*, 2020. 5

[71] Yang Zhao, Chunyuan Li, Ping Yu, Jianfeng Gao, and Changyou Chen. Feature Quantization Improves GAN Training. *arXiv:2004.02088 [cs, stat]*, 2020. 6

[72] Zhengli Zhao, Sameer Singh, Honglak Lee, Zizhao Zhang, Augustus Odena, and Han Zhang. Improved Consistency Regularization for GANs. In *AAAI*, 2021. 1

[73] Brady Zhou and Philipp Krähenbühl. Don't let your Discriminator be fooled. In *International Conference on Learning Representations*, 2018. 5

[74] Peng Zhou, Lingxi Xie, Xiaopeng Zhang, Bingbing Ni, and Qi Tian. Searching towards Class-Aware Generators for Conditional Generative Adversarial Networks. *arXiv:2006.14208 [cs]*, 2020. 2